EVERTON FLÁVIO RUFINO SEÁRA

UMA ARQUITETURA OAI PARA PRESERVAÇÃO DIGITAL UTILIZANDO REDES *PEER-TO-PEER* ESTRUTURADAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Marcos Sfair Sunye

CURITIBA

2008

EVERTON FLÁVIO RUFINO SEÁRA

UMA ARQUITETURA OAI PARA PRESERVAÇÃO DIGITAL UTILIZANDO REDES *PEER-TO-PEER* ESTRUTURADAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Marcos Sfair Sunye

CURITIBA

2008

SUMÁRIO

LISTA DE FIGURAS

RESUMO

A Iniciativa Open Archives (OAI) permite que bibliotecas e museus criem e compartilhem suas próprias Bibliotecas Digitais (BD) com baixo custo. A concepção de BDs OAI é baseada no protocolo OAI-PMH, que, embora tenha se consolidado como padrão para disseminação de metadados, não se preocupa com aspectos relativos à preservação digital e disponibilidade dos objetos, requisitos essenciais neste tipo de sistema. Assim, a construção de mecanismos que garantam melhorias neste âmbito, sem acréscimos de custo, torna-se um grande desafio. Devido suas estruturas topológicas, redes Peer-to-Peer são candidatas naturais para resolver este tipo de problema. Esta dissertação apresenta uma arquitetura OAI para preservação digital baseada em redes P2P estruturadas (Tabelas *Hash* Distribuídas). A proposta da arquitetura é manter as características atuais do protocolo OAI-PMH e realizar, de forma transparente ao usuário, a preservação do conteúdo, considerando a importância do objeto para o administrador. A arquitetura proposta, bem como seus componentes e trocas de mensagens foram validadas através da execução de experimentos e da implementação de protótipos não funcionais.

ABSTRACT

The Open Archives Initiative (OAI) allows both Libraries and Museums creating and sharing their own low-cost Digital Libraries (DL). The conception of OAI DLs is based on OAI-PMH protocol, which is consolidated as a pattern for disseminating metadata, but does not rely on digital preservation and availability of content, essential requirements in this kind of system. So to build new mechanisms to guarantee improvements in this scope, with no cost increases, becomes a great challenge. To solve such a problem, Peer-to-Peer networks are natural candidates due to their topological structures. This work proposes an OAI-based architecture for digital preservation over structured P2P networks (Distributed Hash Tables – DHT). The main goal of the proposal architecture is to keep the current OAI-PMH protocol characteristics and perform, in a transparent way, the preservation of digital content considering the importance of object, defined by the system administrator. The architecture, as well as their components and messages, were validated through execution of some experiments and implementation of proof-of-principle prototypes.

CAPÍTULO 1

INTRODUÇÃO

De acordo com Maly *et. al* [34], o *World Wide Web* encontra-se em sua segunda fase. Na primeira fase, documentos digitais eram disponibilizados utilizando HTML [18] e apresentados através de navegadores. Com o intuito de fornecer melhor suporte à infra-estrutura e gerenciamento das informações, a *World Wide Web Consortium* (W3C) desenvolveu o *Extensible Markup Language* (XML) [6], que permite separar a estrutura do documento de sua representação através da definição de *tags*, que podem representar a estrutura abstrata de um documento.

O advento do XML proporcionou que Bibliotecas Digitais crescessem de forma geométrica na última década, e se tornassem cada vez mais importantes para a sociedade. A Federação de Bibliotecas da Iniciativa *Open Archives* (OAI) [12] pode ser citada como exemplo de como Bibliotecas Digitais têm crescido tanto em número quanto em volume de informação.

Segundo os dados registrados nos repositórios OIA-*compliant* (OAIster), somente no ano de 2006, houve um crescimento de 25% no número de repositórios digitais (subindo para 726) e de 59% no que diz respeito à quantidade de registros armazenados nestes repositórios (subindo para 9.931,910) [32]. Atualmente o número de repositórios ultrapassa os 1000, provendo mais de 17 milhões de registros.

Para fornecer um padrão e facilitar a disseminação de acesso ao conteúdo na Internet, surge o *Open Archives Initiative Protocol for Metadata Harvesting* (OAI-PHM)[24], que é um protocolo baseado em XML com o objetivo de realizar a separação entre os objetos digitais e suas informações descritivas, chamadas de metadados.

No OAI-PMH, uma Biblioteca Digital – conhecida como Provedor de Dados (*Data Provider* – DP) – é responsável pelo armazenamento dos objetos digitais e por expor os metadados destes objetos na Internet. Além disso, um Provedor de Serviço (*Service*

Provider – SP) utiliza-se do protocolo para coletar os metadados (descritos em padrão Dublin Core [3]) a partir de diversos DPs e fornecer um ponto central de busca sobre os objetos.

Embora o protocolo OAI-PMH garanta, de forma eficaz, a disseminação de acesso ao conteúdo, não há preocupação, por parte do protocolo, com a preservação do conteúdo. Assim, falhas nos repositórios podem tornar os materiais indisponíveis, ou fazer com que os mesmos sejam permanentemente perdidos.

Neste contexto, considerando a importância das Bibliotecas Digitais OAI, evidencia-se que a criação de mecanismos confiáveis para realizar preservação de conteúdo nessas instituições torna-se uma necessidade, e um grande desafio, uma vez que a preservação de um objeto implica em sua conservação por longos períodos de tempo – décadas ou séculos.

Devido suas estruturas topológicas, redes Peer-to-Peer (P2P) são candidatas naturais para resolver problemas relacionados à preservação digital. De forma geral, uma rede P2P pode ser formada sem a necessidade de computadores poderosos, tendo em vista que adota como abordagem o gerenciamento descentralizado de recursos computacionais, utilizando-se da distribuição de armazenamento e processamento entre os nós da rede.

Assim, a cooperação entre bibliotecas digitais de diversas instituições através de uma rede P2P pode resultar na criação de um sistema de preservação digital de grande escala, altamente disponível e com baixo custo de concepção e manutenção [26].

Várias abordagens para construir Federações de Bibliotecas Digitais sobre redes P2P têm sido apresentadas com o objetivo de incrementar as capacidades de busca e diminuir o custo de armazenamento do conteúdo [19, 20, 21, 60]. Com relação à preservação digital, também pode-se encontrar na literatura algumas soluções baseadas em redes P2P, como as ferramentas LOCKSS [41] e BRICKS [54], apresentadas no Capítulo 5.

A motivação para o desenvolvimento deste trabalho dá-se pois as soluções para preservação digital existentes não são compatíveis com o protocolo OAI-PMH. Sendo assim, a utilização de tais soluções não permite que exista integração entre Bibliotecas Digitais OAI, a fim de criar um sistema de preservação digital que, além de garantir a confiabilidade

da informação, conserve o modelo de disseminação de metadados fornecido pelo protocolo OAI-PMH.

Além disso, as ferramentas existentes assumem que todos os objetos do sistema possuem a mesma importância, do ponto de vista de preservação. Tendo em vista que, em sua grande maioria, soluções de preservação digital são baseadas em replicação, é importante identificar os objetos que devem ser replicados com maior prioridade, ou seja, possuir mais réplicas no sistema.

Considerar uma única estratégia de replicação para todos os objetos cria um grande número de réplicas desnecessárias no sistema, uma vez que não há bom aproveitamento do espaço em disco. Assim, de maneira geral, tal solução impossibilita a otimização da redução de custo com relação à mídias de armazenamento.

Neste sentido, acredita-se que a criação de uma nova abordagem para preservação digital baseada em redes P2P, que identifique a importância dos objetos no sistema e se integre ao protocolo OAI-PMH, pode convergir para uma solução de baixo custo, que aumente a confiabilidade e disponibilidade do conteúdo, e garanta a visibilidade da informação na Internet.

1.1 Objetivo do Trabalho

Considerando o contexto apresentado, o objetivo geral deste trabalho é conceber uma arquitetura para preservação digital baseada em redes P2P, que possibilite a criação de sistemas de baixo custo e que aumente a confiabilidade e a disponibilidade dos objetos em Bibliotecas Digitais OAI.

Os objetivos específicos são:

1. Definir e descrever os componentes necessários para compor a arquitetura;
2. Projetar a arquitetura de preservação digital;
3. Validar a arquitetura através da implementação de protótipos não funcionais.

1.2 Organização

Esta dissertação está estrutura na seguinte ordem:

O **Capítulo 2** contextualiza o leitor com relação à Bibliotecas Digitais e apresenta a Iniciativa *Open Archives*, abordando aspectos relativos ao protocolo OAI-PMH.

O **Capítulo 3** descreve os conceitos de preservação digital, explicando o que são sistemas de preservação digital e suas possíveis falhas. Este capítulo também apresenta as estratégias de preservação utilizadas neste tipo de sistema e demonstra o funcionamento padrão de uma rede de preservação digital.

Uma fundamentação teórica sobre Sistemas *Peer-to-Peer* é exposta no **Capítulo 4**. Nela são apresentadas as classificações de aplicações e descrito em detalhes dois tipos de arquiteturas P2P: estruturada; e não-estruturada.

O objetivo do **Capítulo 5** é apresentar os principais trabalhos relacionados (LOCKSS e BRICKS) e realizar uma breve comparação com a arquitetura proposta.

O **Capítulo 6** descreve a arquitetura OAI para preservação digital proposta neste trabalho, bem como seus componentes, comportamentos e trocas de mensagem.

Os experimentos realizados para validar a arquitetura são descritos no **Capítulo 7**. E, por fim, as conclusões e trabalhos futuros são apresentados no **Capítulo 8**.

CAPÍTULO 2

BIBLIOTECAS DIGITAIS

Pode-se dizer que – de maneira geral – bibliotecas são essenciais para o funcionamento de uma sociedade democrática, uma vez que são uma grande ferramenta de sabedoria e repositórios de cultura[1].

Tendo em vista que uma biblioteca pode conter uma coleção detalhada sobre um assunto em particular, ou ainda conter coleções detalhadas sobre diversos assuntos, existe uma necessidade inerente de preservar os dados contidos nestas entidades.

Uma Biblioteca Digital (BD) fornece acesso a coleções de conteúdos em meio digital. Em uma BD o material é acessado *on-line*, sem a necessidade de armazenamento físico. Assim, o leitor pode obter acesso ao material com agilidade, independente de sua localidade.

Além disso, em Bibliotecas Digitais materiais podem ser disponibilizados individualmente (livros, teses, dissertações, manuais, etc.) ou em forma de periódicos – caracterizados por agrupar diversas obras, geralmente de diferentes autores, em uma única publicação.

Neste sentido, no que diz respeito ao acesso e disseminação *on-line* de conteúdo, considera-se fundamental a concepção de Bibliotecas Digitais compatíveis com o protocolo OAI-PMH [24]. No entanto, como ocorre em bibliotecas físicas, além de prover acesso ao conteúdo, é necessário que BDs preservem seus "acervos" para que gerações futuras tenham fácil acesso e possam utilizá-lo para pesquisas, aprendizado e ensino. Desta forma, se faz necessário a concepção de mecanismos robustos e baratos que possam controlar e assegurar acesso ao conteúdo por um longo período de tempo [46].

Neste contexto, BDs têm seu conteúdo preservado em formato digital (*bits*). A preservação digital de conteúdo tem por objetivo fornecer subsídios para manter o ma-

[1]Franklin Delano Roosevelt

terial acessível e garantir que seu conteúdo possa ser analisado e interpretado durante longos períodos de tempo. Além disso, a preservação digital tem a função de proteger o conteúdo contra danos e destruição [46].

A preservação dos objetos – de forma geral – é realizada através da replicação de conteúdo em repositórios digitais, e pode ser dividida em duas diferentes abordagens [46]:

- **Centralizada**: A abordagem centralizada caracteriza-se principalmente pela existência de um pequeno número de repositórios robustos e altamente controlados. Nesta abordagem, cada repositório é responsável por realizar todo trabalho de preservação. Os repositórios requerem *hardware* potente e alto grau de suporte técnico, conseqüentemente, altos custos para manutenção são necessários. Além disso, este tipo de abordagem enfatiza a preservação dos objetos (*bits*), deixando questões de acesso a informação em segundo plano;

- **Descentralizada**: Na abordagem descentralizada o conteúdo é armazenado em um grande número de repositórios "livres", ou seja, com controle escasso. Nesta abordagem, cada repositório é responsável apenas por uma parte da preservação, e o custo com *hardware* é relativamente baixo. Além disso, o conteúdo é constantemente analisado, com o intuito de identificar danos. Outro ponto importante é que a preservação é realizada exclusivamente pela Biblioteca Digital, que assume todos os custos. No entanto, os custos são divididos entre as Bibliotecas que participam do processo de preservação, uma vez que o conteúdo é compartilhado entre repositórios de diferentes bibliotecas. Vale ressaltar que, além da preservação dos *bits*, esta abordagem tem como objetivo a preservação de acesso a informação.

Conforme dito no Capítulo 1, novas arquiteturas para construir Federações de Bibliotecas Digitais têm sido propostas, no entanto, enfatizam a melhoria nas capacidades de busca e redução de custo de armazenamento, desconsiderando – de maneira geral – aspectos concernentes à preservação permanente de conteúdo. Sendo assim, este trabalho propõe uma arquitetura OAI para preservação digital baseada em redes P2P estruturadas, com o intuito de garantir preservação digital e aumentar a disponibilidade do conteúdo

em Bibliotecas Digitais OAI.

Com o objetivo de apresentar em detalhes o padrão de disseminação de conteúdo adotado em nossa arquitetura, nas Seções subseqüentes dá-se ênfase a questões pertinentes a Iniciativa *Open Archives* e ao protocolo OAI-PMH.

2.1 Iniciativa *Open Archives*

Desde que a idéia de "auto arquivamento" pessoal de documentos (publicados ou não) foi lançada em 1992, com o sistema arXiv [1], alguns softwares foram criados para evitar que cientistas expusessem seus trabalhos diretamente na Internet sem nenhum tipo de organização [2, 5, 17]. A necessidade de fornecer um sistema de busca centralizado sobre os diversos repositórios individuais de dados disponíveis, conduziu para a criação do OAI *framework*, proposto em 2001.

A Iniciativa *Open Archives* (OAI) [12] desenvolve e promove padrões de interoperabilidade, com o objetivo de facilitar a disseminação eficiente de conteúdo [38]. Uma federação OAI é claramente baseada na separação entre os Provedores de Dados (*Data Providers* – DP) e os Provedores de Serviço (*Service Providers* – SP), conforme pode ser observado na arquitetura apresentada na Figura 2.1.

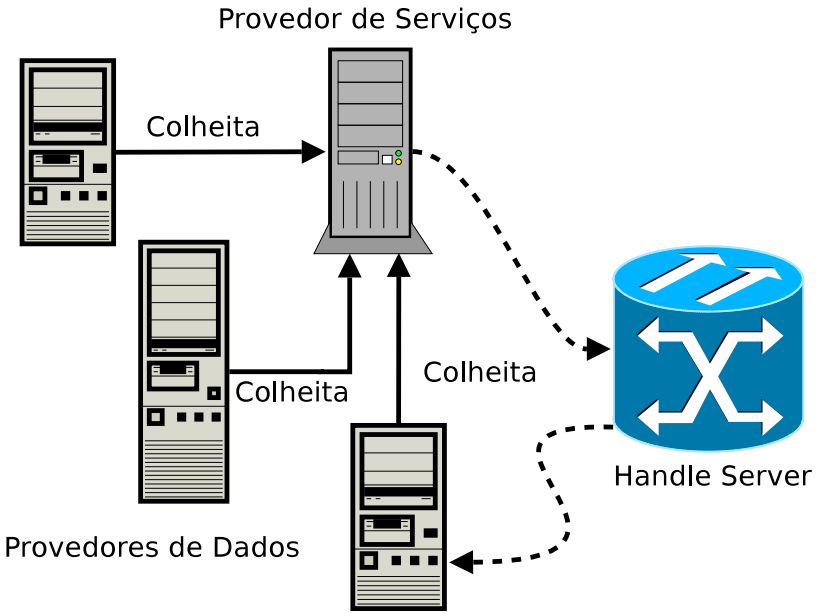

Figura 2.1: Arquitetura de uma Federação OAI

Os Provedores de Dados são aplicações que possuem a responsabilidade de gerenciar os repositórios de conteúdo, onde os objetos digitais (artigos, dissertações, teses, entre outros) são armazenados. Além disso, DPs implementam o protocolo OAI-PMH com o intuito de criar e expor os metadados (informações descritivas sobre os registros dos repositórios) de seus objetos [24].

Provedores de Dados podem ser concebidos com suporte nativo ao protocolo OAI-PMH – através da utilização de *softwares* desenvolvidos para apoiar a Iniciativa *Open Archives*, como DSpace [2] e Eprints [5] – ou ser criados sobre repositórios de dados já existentes, através da implementação de uma interface com o protocolo, como é o caso do arXiv [1] e SciELO [13].

A inserção de um objeto digital no DP é realizada em três passos: (i) um metadado é criado pelo DP seguindo o padrão *Dublin Core* [3]; (ii) um Identificador Permanente de Objeto (*Digital Object Identifier* - DOI) é criado e inserido no metadado; e (iii) o *upload* do objeto é realizado e seu conteúdo armazenado no repositório.

Uma vez que os objetos digitais estão armazenados nos repositórios e seus respectivos metadados disponíveis, surge a figura dos Provedores de Serviço, que são responsáveis por realizar a "colheita" (*harvesting*) de todos os metadados em um conjunto de DPs, respeitando os critérios de coleta disponíveis no protocolo OAI-PMH. Após colhidos, os metadados são armazenados localmente, organizados e utilizados como base para disponibilizar um mecanismo de busca unificado sobre os registros dos repositórios. Desta forma, ao acessar um SP e realizar uma busca, o usuário é direcionado ao DP responsável pelo objeto.

Os metadados podem ser colhidos pelos SP observando dois critérios existentes no protocolo OAI-PMH [24, 38]:

- **Baseado em Data** (*Data-based*): Todos os metadados incluídos ou alterados após a data especificada são colhidos; e
- **Baseado em Conjuntos** (*Set-based*): Todos os metadados que pertencem ao conjunto especificado são colhidos. Nesse contexto, um conjunto é uma estrutura, opcional, para agrupar itens em um repositório.

Para que o SP possa realizar a coleta dos metadados existem seis verbos de requisição do protocolo. Cada verbo atende um tipo específico de solicitação (*request*) e embora a resposta de cada um deles possua um esquema diferente, todas são entregues em formato XML, como pode ser observado no exemplo apresentado na Figura 2.2. A seguir são apresentados os verbos existentes no protocolo [24]:

- *Identify*: Retorna um conjunto das principais informações do repositório. Pode-se citar como exemplo o nome do objeto, identificador, e-mail do administrador e informações sobre a propriedade intelectual dos objetos;
- *ListMetadataFormats*: Recupera os formatos de metadados existentes no DP. Conforme apresentado anteriormente, o formato padrão é o *Dublin Core*;
- *GetRecord*: A partir de um identificador, retorna seu registro correspondente;
- *ListRecords*: Realiza a "colheita" dos metadados do repositório. Pode-se estabelecer coleta seletiva dos objetos através de parâmetros opcionais disponíveis no protocolo;
- *ListIdentifiers*: Similar ao verbo *ListRecords*, no entanto, retorna apenas os identificadores dos registros no repositório;
- *ListSets*: Lista em formato hierárquico os assuntos que classificam os documentos no repositório.

Uma vez que uma federação de bibliotecas OAI é composta por diversos Provedores de Dados, existe grande possibilidade de que ocorram mudanças inesperadas em algum destes provedores (e.g. alterações de nome no servidor, troca de domínio, etc.), acarretando na indisponibilidade de certos objetos digitais, já que os metadados colhidos anteriormente pelo SP permanecem inalterados.

Neste sentido, para que exista a possibilidade de manter os objetos digitais acessíveis mesmo com mudanças nos DPs, irrompe a figura do *Handler Server* (HS). De forma geral, o HS (e.g. HDL [16] ou ACM-DOI [15]) age como um Servidor de Nomes de Domínios (DNS), mapeando o identificador único e permanente de objeto digital (DOI) para seu endereço físico em um dos DPs, conforme ilustrado na Figura 2.1.

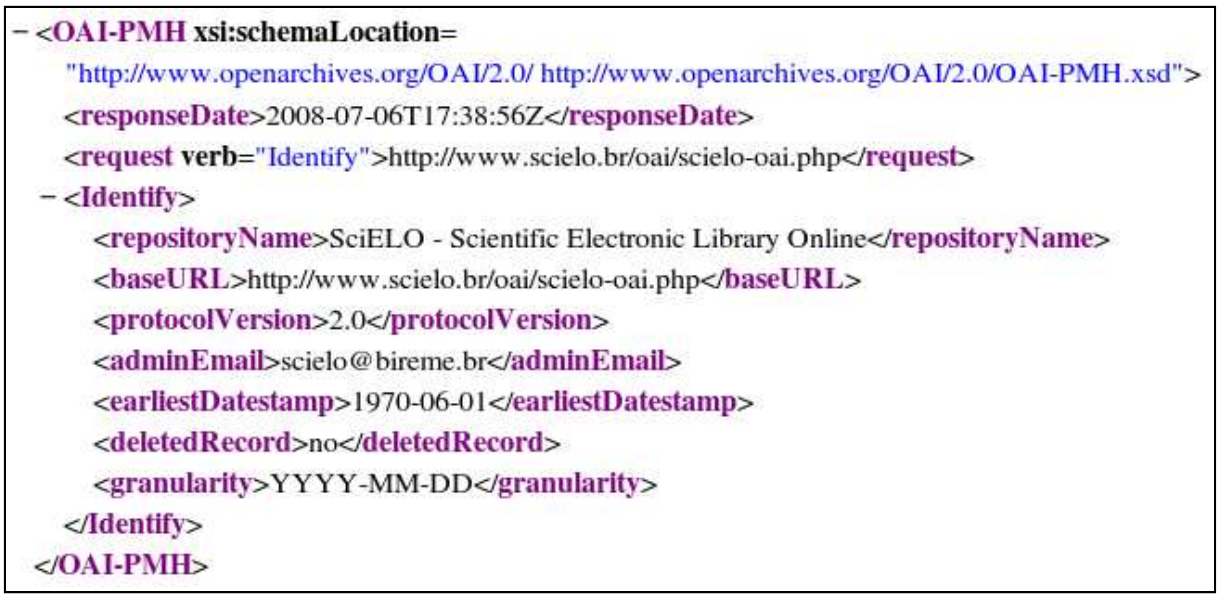

Figura 2.2: Exemplo de resposta XML fornecido pelo DP SciELO [13] a partir do verbo *Identify*

Observando a estrutura de um identificador, por exemplo *hdl.handle.net/1884/7544*, pode-se extrair dados referentes ao mapeamento do objeto. Neste exemplo, o número 1884 (chamado de prefixo) identifica o DP e o número 7544 identifica o Objeto Digital. Sendo assim, a partir do identificador o HS (acessível pela URI *hdl.handle.net*) realiza o mapeamento para o DP correto (neste caso *dspace.c3sl.ufpr.br*, identificado pelo número 1884) e recuperar o objeto requisitado. Esta abordagem permite que o domínio de uma Biblioteca Digital (Provedor de Dados) seja modificado sem a necessidade de mudança nos metadados de seus objetos.

2.1.1 Disponibilidade e Confiabilidade no OAI-PMH

A popularidade do protocolo OAI-PMH entre as Bibliotecas Digitais mais recentes é baseada em sua simplicidade e arquitetura de baixo custo. Outro ponto que colabora, é a existência de *softwares open source* que o implementam, como DSpace [2] e Eprints [5].

Dentre os diversos tipos de Bibliotecas Digitais que formam uma federação OAI, pode-se encontrar pequenos servidores com submissões pessoais, bibliotecas universitárias (principalmente com teses e dissertações), imagens de museus, etc.

Embora federações OAI sejam de suma importância para distribuição de conteúdo

pela Internet, a arquitetura atual não aborda aspectos pertinentes à confiabilidade e disponibilidade dos objetos digitais.

Do ponto de vista da disponibilidade, considera-se o papel desempenhado pelo HS de extrema importância para endereçar, de forma única, objetos armazenados em Bibliotecas Digitais OAI. No entanto, pode-se dizer que o HS caracteriza um ponto central de falha e gargalo na rede, uma vez que todas as requisições de objetos passam, obrigatoriamente, por ele antes de chegarem ao DP. Desta forma, uma falha no HS pode tornar todo sistema indisponível, mesmo que os DPs estejam funcionando perfeitamente.

Um problema ainda maior é a impossibilidade de armazenar um objeto digital em mais de um DP, o que compromete drasticamente a confiabilidade e disponibilidade do objeto. Esta asserção pode tornar o material momentaneamente indisponível, em caso de falha do DP, ou acarretar na perda permanente do conteúdo, caso este seja corrompido ou perdido e não existam mecanismos externos de *backup*.

Neste contexto, evidencia-se a extrema importância da criação de instrumentos que preservem o conteúdo armazenado em Bibliotecas Digitais OAI. É importante ressaltar que tais instrumentos devem ser de baixo custo, devido a grande diversidade das Bibliotecas que compõem uma federação OAI, e preservar os dados por longos períodos de tempo (décadas ou séculos).

Para resolver os problemas supracitados, este trabalho propõe uma arquitetura OAI para preservação digital, descrita no Capítulo 6, baseada fortemente na colaboração entre os Provedores de Dados OAI. A arquitetura proposta define a criação de uma Rede P2P estruturada entre os DPs para realizar a preservação dos objetos, alcançando assim uma solução de baixo custo e que mantém os materiais preservados e disponíveis por longos períodos de tempo.

Com o objetivo de situar o leitor com relação à preservação digital, o próximo Capítulo apresenta os principais conceitos referentes à este assunto.

CAPÍTULO 3

PRESERVAÇÃO DIGITAL

A importância dos objetos digitais é cada vez maior na sociedade atual, desta forma, considera-se como prioridade a preservação de acesso, confiabilidade e integridade dos objetos digitais. Como objeto digital define-se qualquer informação que possa ser representada através de uma seqüência de *bits* [55]. Alguns exemplos clássicos são documentos de texto, planilhas eletrônicas, imagens, vídeos, músicas, páginas *Web* etc.

Neste sentido, Preservação Digital (PD) pode ser observada como um conjunto de atividades ou processos responsáveis por garantir o acesso continuado a longo prazo aos objetos digitais e a todo o patrimônio cultural existente em formato digital [58]. Neste tipo de contexto, o tempo em que as informações devem ser preservadas torna-se um problema chave, uma vez que mídias de armazenamento e componentes de *hardware* e *software* possuem tempo de vida pré-determinados. Além disso, deve ser possível interpretar a informação independente do formato em que esta foi codificada [48].

Uma vez que este trabalho propõe uma arquitetura OAI para preservação digital de conteúdo, nas seções subseqüentes dá-se ênfase à questões relacionadas à Sistemas de Preservação Digital, bem como a suas estratégias de preservação e acesso ao conteúdo.

3.1 Sistema de Preservação Digital

De acordo com Lu e Chiueh (2006) [39], um Sistema de Preservação Digital pode ser definido como um sistema capaz de preservar dados imutáveis por um longo período de tempo.

O *Open Archival Information Systems* (OAIS) é um modelo referência para sistemas de preservação digital desenvolvido pelo *Consultative Committee for Space Data Systems* (CCSDS) a pedido da *International Organization for Standardization* (ISO) [29]. De maneira geral o modelo OAIS tem por objetivo definir um conjunto de recomendações

e padrões para manter informações armazenadas por longos períodos de tempo (*Long Term*).

Por *Long Term* entende-se o tempo necessário para manter a informação preservada, independente de qualquer impacto sobre mudanças tecnológicas (incluindo suporte a novas mídias e formatos de dados) ou alteração no perfil do público alvo do conteúdo [29]. Desta forma, o tempo de preservação pode se estender indefinidamente, podendo-se considerar décadas, ou até mesmo séculos.

O modelo OAIS é formado por seis entidades funcionais que interagem com três diferentes atores, conforme ilustra a Figura 3.1. Cada ator possui um papel específico no modelo: (i) produtor (*producer*), responsável pela submissão da informação armazenada no sistema (um produtor pode ser um elemento externo ou o administrador do sistema); (ii) administrador (*management*), responsável por estabelecer políticas de preservação e monitorar o sistema; e (iii) consumidor (*consumer*), ator que acessa o sistema em busca de informações. Vale destacar que as entidades funcionais fornecem subsídios para o funcionamento do sistema visando atender os três níveis de usuários mencionados.

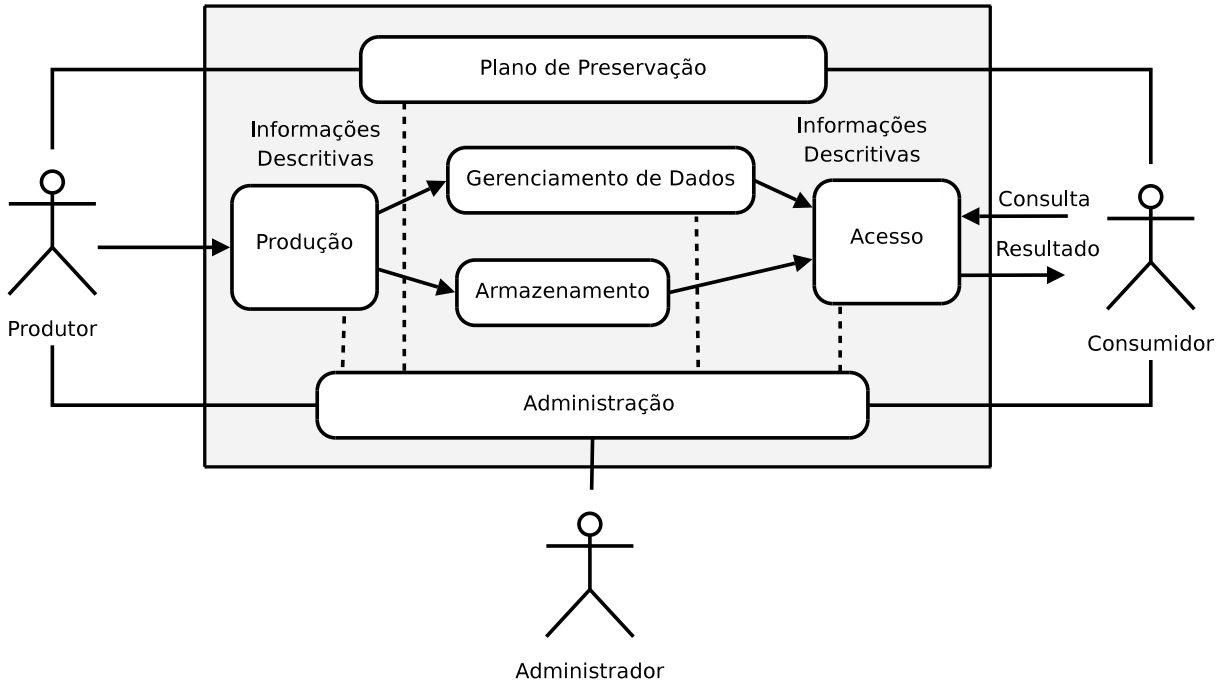

Figura 3.1: Modelo de Atores Entidades Funcionais OAIS. Adaptado de [29]

É de suma importância esclarecer que sistemas de preservação digital possuem carac-

terísticas que os diferenciam dos sistemas de armazenamento tradicionais. Sistemas desta categoria devem ser baratos de construir e manter. Além disso, embora seja desejável, não precisam necessariamente operar de maneira rápida, em contrapartida, devem funcionar corretamente durante longos períodos de tempo, mesmo diante de situações adversas. Outra característica de extrema importância em um sistema de preservação digital é que seus objetos não devem ser apagados ou atualizados.

Com relação ao armazenamento, como apresentado no Capítulo 2, em sistemas de preservação digital informações são armazenadas em dois tipos de repositórios: (i) centralizado; e (ii) descentralizado, sendo o segundo utilizado como base para propor a arquitetura deste trabalho.

Com o intuito de garantir confiabilidade do conteúdo, três *atributos de segurança* são ser considerados [39]:

1. **Confidencialidade**: Indica que objetos digitais não podem ser lidos por usuários não autorizados. Confidenciabilidade também implica na criptografia dos objetos digitais com o objetivo de prevenir ataques;

2. **Integridade**: Garante que os objetos digitais permaneçam consistentes sob modificações inesperadas, ataques maliciosos, falhas catastróficas ou obsolescência tecnológica;

3. **Autenticidade**: Garante a identidade do objeto digital, através da validação de sua integridade. A autenticidade deve prover mecanismos de auditoria sobre a integridade dos objetos digitais.

Para garantir os atributos de segurança, sistemas de preservação digital devem tratar e, possivelmente, prever um conjunto de falhas que podem ameaçar o funcionamento do sistema, conforme apresenta a Seção 3.2.

3.2 Falhas em Sistemas de Preservação Digital

Como qualquer outro sistema computacional, sistemas de PD estão sujeitos a diversos tipos de falhas. Além de herdarem as possíveis falhas aplicadas à sistemas de arquivamento

digital de curto prazo, possuem falhas características a sua categoria. Desta forma, falhas em sistemas de preservação digital podem ser caracterizadas da seguinte forma, conforme apresentado em [48]:

- **Problemas na mídia**: Mídias de armazenamento são componentes críticos para sistemas de arquivamento e estão sujeitas a deterioração gradual dos dados, podendo causar danos irreversíveis;
- **Falhas no *hardware***: Componentes de *hardware* estão sujeitos à falhas transientes, como falta de energia elétrica, ou a falhas catastróficas permanentes, como por exemplo a destruição do equipamento;
- **Falhas no *software***: Softwares podem apresentar comportamentos anormais que comprometam a integridade dos conteúdos armazenados;
- **Erros de comunicação**: Sistemas de PD devem assumir que a rede utilizada para inserir e disseminar conteúdo pode falhar a qualquer momento (e.g. não entregar um pacote) e provocar inconsistências nos objetos do sistema;
- **Falha nos serviços de rede**: Sistemas de PD devem estar cientes e tentar se proteger contra falhas transientes ou irreversíveis a serviços rede utilizados, como por exemplo, URIs persistentes ou DNS;
- **Obsolescência de *hardware* e mídia**: Mídias e componentes de *hardware* podem eventualmente falhar. Além disso, podem se tornar obsoletos e não conseguir se comunicar com novos componentes de *hardware* adicionados ao sistema;
- **Obsolescência de *software***: Objetos digitais podem ser codificados por formatos obsoletos, impossibilitando sua decodificação e, conseqüentemente, sua interpretação de maneira legível;
- **Falha humana**: Administradores podem, acidentalmente, apagar conteúdos importantes ou ainda interromper o sistema através de alguma ação errônea;

- **Desastre natural**: Desastres como terremotos, furações, inundações, ou incêndios devem ser considerados. A alocação do conteúdo em locais geograficamente separados (através de replicação) pode resolver este problema;

- **Ataques externos e internos**: Sistemas de PD devem considerar que estão vulneráveis à ataques maliciosos vírus ou *worms*. E ainda que, por questões econômicas ou emocionais, administradores (pessoas autorizadas) podem danificar o sistema;

- **Falha econômica**: A informação em formato digital é mais vulnerável a permanecer indisponível por restrições orçamentárias, como por exemplo, o não pagamento de taxas de energia elétrica, conexão de Internet/rede, aquisição de novos componentes de *hardware*. O orçamento necessário para manter o conteúdo preservado e, eventualmente, disponível deve ser previsto;

- **Falha organizacional**: O sistema deve considerar que o conteúdo digital deve ser preservado independente de alterações no rumo ou na missão da organização.

Para cada uma das falhas apresentadas – que podem ameaçar o funcionamento correto do sistema – é necessário observar a relação *custo* x *benefício*, e buscar o melhor benefício dentro do orçamento existente [48].

3.3 Estratégias de Preservação

Diversas estratégias podem ser utilizadas para manter o conteúdo digital preservado. Estratégias baseadas em replicação têm o intuito de obter redundância da informação, preservando as seqüências de *bits* e protegendo o conteúdo contra ataques, falhas em componentes de *hardware*, desastres naturais, etc.

Algumas estratégias, como migração e emulação, são comumente utilizadas com o objetivo de solucionar problemas relativos à obsolescência do formato de codificação. Além disso, mecanismos de auditoria garantem a autenticidade e integridade dos objetos digitais.

Um sistema de preservação digital deve combinar, com transferência, diferentes estratégias para manter seus objetos digitais seguros e preservados. Neste sentido, as próximas seções descrevem as principais estratégias de preservação no âmbito dos sistemas de preservação digital.

3.3.1 Replicação

Pode-se dizer que a informação Digital é mais bem preservada através da sua replicação em múltiplos repositórios executados por organizações autônomas. Essa é a forma encontrada por bibliotecas físicas para preservar seus conteúdos e que tem sido estendida para preservação de dados digitais [27].

A criação e manutenção de múltiplas cópias de um objeto (chamadas de réplicas) em sítios distintos recebe o nome de replicação [50]. Uma única cópia de um conteúdo em uma única localidade torna-se altamente vulnerável às falhas detalhadas na Seção 3.2. Desta forma, considera-se que a possibilidade dos conteúdos digitais não serem perdidos é maior se estes estiverem replicados em diversas localidades [41].

Embora a replicação de conteúdo introduza uma série de dificuldades com relação à migração e controle de acesso, visto que o sistema deve estar apto à gerenciar diversas cópias do mesmo conteúdo, esta estratégia é de suma importância em sistemas distribuídos, visto que: (i) aumenta a disponibilidade dos objetos; (ii) melhora o desempenho do sistema, uma vez que diversos sítios podem atender requisições referentes ao mesmo objeto digital; e (iii) contribui para o incremento da confiabilidade dos dados.

Normalmente objetos digitais (neste caso uma seqüência de *bits*) são replicados de forma completa, ou seja, o conteúdo do objeto é replicado como um todo para outro sítio. Outra alternativa para criar redundância de objetos é a técnica *Erasure Code*, que gera diversos fragmentos do conteúdo, com o intuito de diminuir a sobrecarga de armazenamento e banda de rede [57]. No entanto, devido sua complexidade de implementação e manutenção, sua aplicação em sistemas de larga escala é praticamente inexistente.

É importante salientar que no âmbito da preservação digital não há atualização das réplicas, umas vez que os objetos são imutáveis [42]. Desta forma, cada réplica criada

tem sua preservação garantida ao longo do tempo.

3.3.2 Migração

Pode-se dizer que migração é um conjunto de atividades que desempenham o papel de transferência de dados para um novo ambiente. Migrações podem ser consideradas como eventos excepcionais ou rotineiros, controlados por administradores de sistema ou executados automaticamente, sem intervenção humana [48].

Do ponto de vista da preservação digital, migração pode ser utilizada para transferir conteúdo de uma mídia de armazenamento antiga para um novo dispositivo. No entanto, esta estratégia é comumente utilizada para realizar migração de formato (por exemplo, a conversão de um documento *Microsoft Word* para um arquivo PDF ou *OpenDocument Format* – ODF).

Um grande problema encontrado na utilização desta técnica são as eventuais perdas de informação durante a execução do processo. Desta forma, alguns sistemas (como por exemplo, a Biblioteca Nacional dos Países Baixos, *Koninklijke Bibliotheek* – KB) evitam a utilização do processo de migração de formato, por aceitar informações para preservação apenas em formatos considerados "satisfatórios", como por exemplo PDF, no caso da KB [48].

3.3.3 Emulação

A emulação é alternativa para suprir a deficiência de acesso à formatos obsoletos, sem a necessidade utilizar técnicas de migração e correr riscos com relação à perda de conteúdo.

A idéia essencial por trás da emulação é ter a habilidade para acessar ou executar um conteúdo específico, em seu formato original, fora de sua plataforma. Para tal, utiliza-se um conjunto de softwares que recriam (emulam) a plataforma onde o objeto foi criado originalmente, possibilitando o acesso legível à informação, sem a necessidade de mudanças de formato [31].

Tendo em vista a dificuldade para construção de ambientes emulados, esta abordagem não é amplamente utilizada.

3.3.4 Auditoria

Define-se auditoria em sistemas de preservação digital como um grupo de funções responsáveis por garantir integridade e autenticidade dos objetos digitais. Estratégias de auditoria são necessárias para detectar objetos corrompidos e, se possível, restaurá-los sem intervenção humana.

Pode-se dizer que, por natureza, grande parte dos objetos digitais armazenados em sistemas de preservação digital são raramente acessados, portanto, eventos gerados a partir dos acessos dos usuários aos objetos não podem ser utilizados para acionar funcionalidades que detectem inconsistências nos objetos. Neste contexto, mecanismos de auditoria devem ser executados frequentemente, para manter a probabilidade de objetos corrompidos dentro dos níveis aceitáveis [48].

De maneira geral, processos de auditoria baseiam-se em *hashs* para identificar falhas. Embora essa técnica seja considerada eficaz, pode ser extremamente lenta para auditar, por completo, mídias com grande potencial de armazenamento. Além disso, o aumento contínuo de capacidade de armazenamento dos discos rígidos faz com que o tempo para realizar auditoria seja cada vez maior [23]. Sendo assim, considera-se necessário a concepção de novas abordagens de auditoria, ou variações da abordagem baseada em *hashs*, para que processos de auditoria não interfiram no desempenho do sistema como um todo.

Neste sentido, sistemas de preservação digital concebidos sobre redes P2P podem aproveitar-se de sua natureza distribuída para conceber mecanismos de auditoria que não interfiram no desempenho do sistema. Um exemplo é o projeto LOCKSS [41], que se utiliza da rede P2P para propor um processo de auditoria baseado em votos. Tal sistema, apresentado na Seção 5.2 é utilizado como base para o mecanismo de auditoria proposto em nossa arquitetura.

3.4 Redes de Preservação Digital

Uma possível solução para concepção de sistemas de PD é a utilização do modelo descentralizado de repositórios, formando assim uma Rede de Preservação Digital baseada em

replicação.

Neste sentido, a estratégia de replicação garante que o conteúdo possa ser recuperado a partir de diferentes localidades, aumentando a confiabilidade e disponibilidade do conteúdo [25]. Além disso, mecanismos de auditoria garantem a integridade das réplicas.

Com o intuito de se determinar um método para medir a confiabilidade dos objetos em uma rede de preservação digital são consideradas duas variáveis [27]: (i) *confiabilidade do sítio*, probabilidade de um sítio não falhar; e (ii) *confiabilidade local*, probabilidade de recuperar um objeto originado em um determinado sítio. Através destas variáveis, pode-se calcular a probabilidade de se recuperar um objeto de uma rede de preservação digital.

Neste sentido, a Figura 3.2 exemplifica uma rede de PD formada por três sítios (**1**, **2** e **3**). Considera-se, neste exemplo, que cada um dos sítios pode falhar de forma independente e que a *confiabilidade do sítio* é de 0.8 para cada um dos nodos. Desta forma, a probabilidade de falha de cada nodo é de 0.2 (20%).

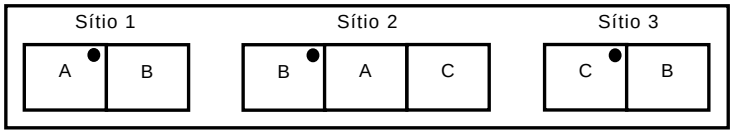

Figura 3.2: Exemplo de confiabilidade dos sítios

Considera-se também os objetos **A**, **B** e **C**, replicados de maneira uniforme entre os nodos da rede. O círculo associado a cada objeto indica seu sítio de origem.

Observando este ambiente pode-se concluir que o objeto **A** será perdido somente se os sítios **1** e **2** falharem. Tendo em vista que probabilidade dos sítios **1** e **2** falharem é *0.2* x *0.2 = 0.04*, então a probabilidade do objeto **A** ser recuperado é *1 - 0.04 = 0.96*, ou seja, sua *confiabilidade local* é 96%.

Além disso, o objeto **B** é aceito como o mais seguro, já que este será perdido somente se os 3 sítios da rede falharem simultaneamente. Desta forma, a probabilidade de recuperar o objeto **B** é de $1 - 0.2^3 = 0.992$ (99.2%).

Uma vez que uma rede de preservação pode ser concebida sobre um sistema P2P, o Capítulo 4 apresenta conceitos relátivos à esse tipo de sistema.

CAPÍTULO 4

SISTEMAS PEER-TO-PEER

Sistemas Peer-to-Peer (ou P2P) adotam uma abordagem descentralizada no que diz respeito ao gerenciamento de recursos computacionais [56]. De forma geral, através da distribuição de armazenamento de dados, processamento e largura de banda entre todos os nodos da rede, eles podem "escalar" sem a necessidade de computadores poderosos e intermediação de um servidor centralizado [22, 56].

Pode-se dizer que a motivação por trás das aplicações baseadas em arquiteturas P2P deriva-se da sua habilidade de funcionar, "escalar" e de se auto-organizar em redes com alta taxa de nodos transitórios e na presença de redes e computadores falhos, mantendo níveis de performance e conectividade aceitáveis [22]. Tais características fazem com que a arquitetura P2P seja utilizada como base em diversos sistemas distribuídos, tais como Gnutella [9], Seti@home [14], Freenet [7], eMule [4], entre outros.

Diversas definições de sistemas P2P podem ser encontradas na literatura, uma delas diz respeito a sistemas P2P "puros", o que caracteriza seu funcionamento como totalmente descentralizado, além de estabelecer equivalência com relação a funcionalidades e execução de tarefas para todos nodos do sistema. No entanto, este tipo de definição não é totalmente abrangente, já que não engloba sistemas que utilizam "supernodos" (nodos que funcionam como mini-servidores, responsáveis por funções específicas e que são alocados dinamicamente), nem sistemas que utilizam algum servidor central para realizar tarefas secundárias [22, 40].

Uma definição amplamente aceita diz que "sistemas P2P são uma classe de aplicações que levam vantagem de recursos computacionais, como armazenamento, processamento e conteúdo disponíveis na internet" [52]. Entretanto, devido seu alto nível de abstração, esta definição engloba outros tipos de aplicações, como por exemplo computação em Grid, e sistemas que contam com servidores centralizados.

De acordo com Androutsellis-Theotokis e Spinellis (2004) [22], a definição de um sistema P2P deve priorizar as seguintes características: (i) compartilhamento de recursos computacionais de forma direta; (ii) servidores centralizados podem ser utilizados apenas para especificar tarefas, não para operações básicas; (iii) habilidade para tratar instabilidade da rede e conectividade variada e; (iv) tolerância a falhas e auto-organização.

Neste contexto, pode-se dizer que sistemas Peer-to-Peer são sistemas distribuídos que consistem de nodos inter-conectados em uma rede, aptos a se auto-organizar, com propósito de compartilhamento de recursos computacionais e capacidade de se adaptar a falhas e nodos transitórios, mantendo conectividade e performance aceitáveis, sem requerer intermediação de um servidor centralizado [22].

Inúmeras categorias de aplicações, arquiteturas e topologias de rede podem ser consideradas em uma rede P2P, sendo assim, as seções subseqüentes têm o intuito de apresentar informações pertinentes a sua classificação (seção 4.1) e arquitetura (seção 4.2).

4.1 Classificação das Aplicações

Redes P2P podem ser empregadas em diversos tipos de aplicações. A seguir são apresentadas as categorias de aplicações, bem como citados – de forma breve – alguns sistemas já existentes que fazem parte de cada categoria.

Comunicação e Colaboração: Nesta categoria incluí-se sistemas que fornecem infraestrutura adequada para realizar comunicação e colaboração direta e em tempo real entre os nodos da rede [22]. Como exemplo, pode-se citar os sistemas de *Instant Messaging*, como ICQ, Skype, MSN Messenger, entre outros.

Computação Distribuída: Os sistemas desta categoria têm por objetivo se beneficiar do poder de processamento (ciclos de CPU) disponível em cada um dos *peers* [22]. Para alcançar tal funcionalidade, grandes tarefas devem ser divididas em unidades menores e cada unidade deve ser executada de forma distribuída entre os diversos *peers* da rede. Vale ressaltar que neste tipo de aplicação algum tipo de coordenação central é necessária, uma vez que é preciso dividir as tarefas e coletar os resultados

da execução dos diversos *peers* participantes. Pode-se citar como exemplos clássicos desta categoria os projetos Seti@home [14] e o Genome@home [8]. Outro exemplo interessante é a implementação do algoritmo de *pagerank* do Google, descrito em [51].

Sistemas de Banco de Dados: Nesta categoria o intuito é projetar sistemas de banco de dados distribuídos com capacidade de funcionar sobre as arquiteturas de rede Peer-to-Peer [22]. Neste contexto, propôs-se o Modelo Relacional Local (ou LRM - *Local Ralational Model*) [44] como um modelo de dados. O LRM assume que o conjunto de todos os dados em uma rede P2P são armazenados em bancos de dados (relacionais) locais inter-conectados e define as regras e dependências semânticas entre eles. Outro exemplo desta categoria é o PIER [33], uma *engine* de busca distribuída e escalável, construída sobre uma rede P2P estruturada e capaz de executar consultas relacionais sobre centenas de computadores.

Distribuição de Conteúdo: Um sistema P2P de distribuição de conteúdo cria uma forma de armazenamento distribuído que permite a publicação, consulta e recuperação de conteúdo pelos membros de uma rede P2P [22]. Atualmente, a maior parte dos sistemas P2P existentes enquadram-se nesta categoria, onde pode-se encontrar desde de aplicações de compartilhamento direto de arquivos, até sistemas mais sofisticados com o intuito de criar uma rede de distribuição de conteúdo segura e com funcionalidades de publicação, organização, indexação, consulta, recuperação e alteração de dados de forma eficiente. Como exemplo desta categoria pode-se citar sistemas como Gnutella [9], Kazaa [10], eMule [4], Freenet [7], Chord [53], entre outros.

4.2 Arquitetura

Uma rede Peer-to-Peer é formada sobre uma rede de computadores, que é tipicamente concebida por computadores (neste contexto, *peers* ou nodos) e suas conexões (*edges*), e desta forma, pode ser considerada uma rede *Overlay*. Sendo assim, a topologia, estrutura

e grau de centralização da rede são cruciais para a operação dos sistemas que a utilizam.

Embora – teoricamente – em sua forma mais pura as redes Peer-to-Peer sejam consideradas totalmente descentralizadas, na prática isto nem sempre ocorre, e sistemas com diferentes tipos de centralização podem ser encontrados [22]. Deste modo as seções subseqüentes têm por objeto apresentar as principais arquiteturas que podem ser utilizadas em uma rede P2P.

4.2.1 Arquitetura Não Estruturada

Em uma arquitetura não estruturada a disposição do conteúdo não possui nenhuma relação com a topologia da rede virtual (*overlay*), desta forma, o conteúdo necessita ser localizado antes de ser recuperado.

Os mecanismos de consulta em uma arquitetura estruturada – de forma geral – baseiam-se em métodos de força bruta, onde mensagens com os termos de consulta são espalhadas pela rede (*flooding*) e as buscas são realizadas em largura ou profundidade, até que o conteúdo desejado seja localizado [22].

Embora seus mecanismos de consulta possam causar diversas complicações, principalmente no que diz respeito a disponibilidade e escalabilidade, pode-se dizer que sistemas P2P baseados em arquiteturas não estruturadas são adequados para redes com alta taxa de nodos transitórios [22].

Tendo em vista que diferentes classificações para arquiteturas não estruturadas podem ser consideradas, no que concerne ao seu nível de centralização e estrutura de rede, a seguir são apresentadas suas diferentes categorizações.

Puramente Descentralizada: A arquitetura puramente descentralizada (também conhecida como "pura") é considerada a mais simples [56] e baseia-se no fato de que não existe coordenação central das atividades na rede. Além disso, em uma arquitetura puramente descentralizada, todos *peers* são considerados clientes e servidores (sendo também chamados de *servents*), e possuem o mesmo papel, no que diz respeito a execução de tarefas. [56, 22]. A Figura 4.1 ilustra a arquitetura de um sistema puramente descentralizado, apresentando claramente a conexão direta entre cada um dos *peers* na rede.

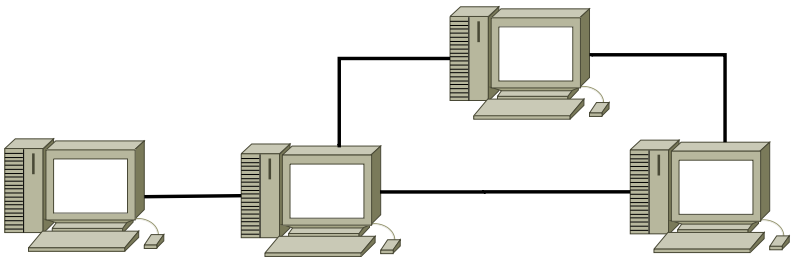

Figura 4.1: Arquitetura Puramente Descentralizada

Pode-se dizer que um dos principais exemplos de sistemas baseados nesta arquitetura é o Gnutella [9]. Isto se dá devido sua arquitetura aberta, escalabilidade e capacidade de auto-organização [22]. Para realizar consultas na rede, o Gnutella utiliza-se de mecanismos de *flooding* (ou *broadcast*), onde cada *peer* recebe mensagens de todos os seus vizinhos. Como resposta, as mensagens retornam ao *peer* solicitante através do caminho inverso [22].

Escalabilidade é uma característica marcante da arquitetura P2P puramente descentralizada. Embora seja discutida – devido seu mecanismo de busca baseado em *flooding* – sua dificuldade de escalar até um número muito grande de *peers* [56], a escalabilidade neste tipo de arquitetura é garantida através do tempo de vida de cada mensagem (TTL, ou *Time To Live*), o que cria uma espécie de sub-rede virtual e faz com que as mensagens não sejam entregues para toda rede [22].

Também é importante destacar que neste tipo de arquitetura dados relevantes podem facilmente encontrar-se indisponíveis, uma vez que os *peers* que o disponibilizam podem estar off-line ou ter falhado. No entanto, como todos os *peers* executam as mesmas funções, estes estão aptos a replicar os dados, gerando assim uma alta tolerância a falhas [56].

Parcialmente Centralizada: Pode-se dizer que a arquitetura parcialmente centralizada é considerada híbrida e possui características tanto de sistemas P2P puros como de sistemas cliente-servidor [56].

Embora os sistemas baseados em arquitetura parcialmente centralizada sejam similares aos sistemas P2P puramente descentralizados, esta arquitetura conta com um *peer* "especial", chamado de *super-peer* ou *supernodo*. Um *super-peer* é um *peer* alocado dinamicamente pelo sistema – a partir da análise da sua largura de banda e poder de

processamento [22] – e tem a função de agir como servidor dedicado para alguns *peers* da rede. Sua principal função é executar tarefas mais complexas, como indexação, processamento de consultas, controle de acesso, gerenciamento de metadados, entre outras [22, 56]. A Figura 4.2 apresenta a arquitetura de uma pequena rede parcialmente centralizada, onde os *peers* conectam-se diretamente a um *super-peer*, que pode estabelecer conexão direta com diversos outros *peers* especiais.

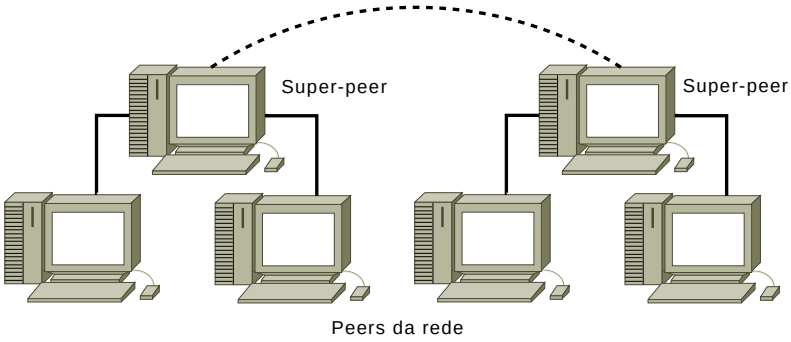

Figura 4.2: Arquitetura Parcialmente Centralizada

De maneira geral, as duas principais vantagens dos sistemas parcialmente centralizados são:

- O tempo de resposta é reduzido, se comparado a sistemas puramente descentralizados, mesmo que não exista um ponto único de falha [22]. Embora um *super-peer* execute o papel de um servidor centralizado, em caso de falha os *peers* conectados a ele podem realizar uma conexão com outros *super-peers*, o que garante a continuidade de operação da rede.

- A heterogeneidade inerente das redes P2P é explorada.

Híbrida Descentralizada: Em uma arquitetura híbrida descentralizada cada *peer* armazena conteúdos que serão compartilhados com o restante da rede [22]. Além disso, conforme apresenta a Figura 4.3, cada um dos *peers* está diretamente conectado a um servidor central, responsável por:

- Manter uma tabela com informações das conexões dos usuários (endereço IP, largura de banda, etc.);

- Manter uma lista dos arquivos armazenados por cada *peer*, indexado através de um metadado que contém a descrição, data e hora de criação do arquivo, etc.

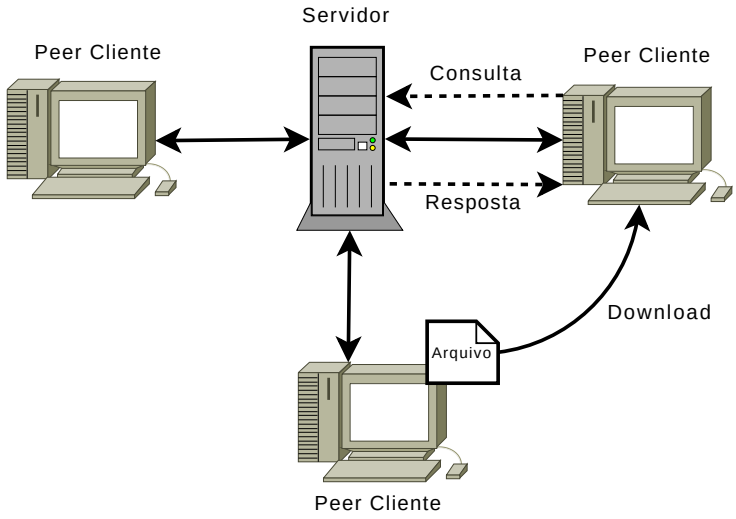

Figura 4.3: Arquitetura Híbrida Descentralizada

Neste tipo de arquitetura, um computador que deseja se juntar a rede, contata o servidor central e informa os arquivos que irá manter. Quanto um *peer* deseja fazer uma busca por um determinado conteúdo, este envia uma requisição ao servidor central, que por sua vez realiza a busca e retorna o endereço do *peer* responsável por manter o arquivo. Então, os *peers* podem comunicar-se diretamente e realizar a transferência dos conteúdos.

Um vantagem dessa abordagem é sua simplicidade de implementação e rapidez e eficiência na localização dos arquivos. No entanto, esse tipo de arquitetura torna-se altamente susceptível a falhas e ataques maliciosos ao servidor central. Além disso, sistemas deste tipo são considerados pouco escaláveis, uma vez que a performance das buscas é altamente comprometida com a entrada de novos membros na rede.

4.2.2 Arquitetura Estruturada

Pesquisas iniciais em sistemas P2P tinham por objetivo principal a melhoria do desempenho dos sistemas baseados em arquitetura não estruturada [56], no entanto, conduziram para a criação de um a nova arquitetura, conhecida como arquitetura estruturada. Em uma arquitetura estruturada a topologia da rede é bem definida e a alocação do conteúdo

determinada pelo sistema, impactando em buscas mais eficientes.

De maneira geral, arquiteturas estruturadas utilizam-se de Tabelas *Hash* Distribuídas (DHT) [45, 49, 53] para realizar suas operações básicas (inserção e busca) e proporcionar a concepção de sistemas escaláveis e auto-gerenciáveis. Como ocorre em tabelas *hash*, uma DHT associa uma chave (*hash*) a um valor (conteúdo) (Figura 4.4) e, posteriormente, utiliza-se da chave para localizar o conteúdo. Vale destacar que uma chave é associada somente a um objeto, e que operações de busca podem ser originadas em qualquer nodo da DHT. Durante a busca os nodos realizam as operações necessárias para encontrar o conteúdo na rede e entregá-lo ao solicitante.

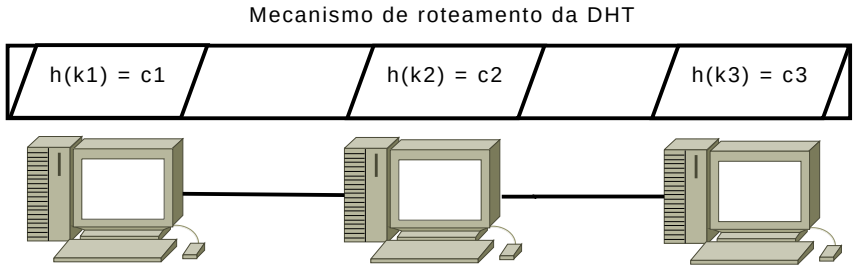

Figura 4.4: Organização do conteúdo em uma DHT. Conteúdo é associado a uma chave e alocado em um nodo da rede.

Tradicionalmente, inserções e buscas de objetos em DHTs são realizadas a partir de duas funções:

- `put(chave, valor)`: utilizada para inserir um objeto na rede, sendo que o parâmetro *chave* é o identificador do objeto e o atributo *valor* seu conteúdo;
- `get(chave)`: recupera um objeto da DHT a partir de sua *chave*.

Diferentes DHTs são encontradas na literatura, cada uma empregando mecanismos distintos para realizar o roteamento de mensagens e localização de dados. Os tópicos a seguir apresentam uma breve introdução sobre os principais protocolos DHT existentes:

- **Chord**: utiliza-se de um *círculo* de identificares para organizar seus sítios e localizar os objetos. Uma rede Chord [53] possui formato circular e suas mensagens são encaminhadas em sentido horário. Desta forma, cada sítio armazena ponteiros para

seus sucessores e predecessores. Assim, espera-se que seja necessário o envio de $O(logN)$ mensagens, em uma rede de N sítios, para localizar um objeto;

- **CAN** (*Content Addressable Network*): CAN [45] utiliza-se de um espaço de coordenadas cartesiano n-dimensional para implementar localização distribuída. Além disso, possui uma tabela de roteamento que associa cada nodo a uma *zona* no espaço de coordenadas;

- **Pastry**: De forma similar ao Chord, o sistema Pastry [49] mantém uma tabela de vizinhos para realizar o roteamento das mensagens e a localização do conteúdo.

Embora DHTs proporcionem boa performance em pesquisas, possuem problemas com sistemas que possuem populações muito transientes, pois é necessário manter as tabelas de roteamento atualizadas a cada entrada e saída de nodos. Além disso, consultas podem ser altamente restritivas, pois o conteúdo pesquisado deve corresponder exatamente ao valor da chave.

É importante destacar que, Tabelas *Hash* Distribuídas são utilizadas para compor a base da estrutura de preservação digital de nossa arquitetura. Tal utilização justifica-se pois buscas por objetos no sistema serão realizas a partir de uma chave única (DOI), e pesquisas complexas são supridas pelos Provedores de Serviço OAI (com base nos metadados colhidos). Além disso, nossa arquitetura é voltada, especialmente, para bibliotecas digitais de larga escala, que de forma geral possuem uma pequena taxa de entrada e saída de nodos.

Antes de descrever a arquitetura proposta nesse trabalho, é conveniente apresentar outras soluções para preservação digital também baseadas em redes P2P. Sendo assim, o Capítulo 5 registra uma breve descrição sobre os trabalhos relacionados.

CAPÍTULO 5

TRABALHOS RELACIONADOS

Literaturas sobre sistemas de preservação digital baseados em redes P2P podem ser consideras escassas, já que a maioria dos trabalhos de armazenamento de dados baseados em redes P2P (por exemplo, Freenet [7] e CFS [28]) utilizam armazenamento de curta duração.

Uma vez que nossa arquitetura é baseada em armazenamento e preservação de longa duração, dois principais trabalhos são evidenciados e apresentados nas Seções subseqüentes.

5.1 O Projeto BRICKS

O projeto BRICKS (*Building Resources for Integrated Cultural Knowledge Services*) tem por objetivo projetar, desenvolver e manter uma infra-estrutura descentralizada baseada em serviços para bibliotecas digitais, com o intuito de compartilhar conhecimento e recursos culturais de domínio público. Neste sentido, a infra-estrutura BRICKS utiliza a Internet como um backbone, para garantir escalabilidade, disponibilidade e interoperabilidade [54].

Além disso, o projeto BRICKS tem o intuito de criar uma rede com baixo custo de manutenção e que considere o mínimo de investimentos necessários, no que diz respeito à infra-estrutura, para a entrada de novos membros. Sendo assim, utiliza-se de uma arquitetura baseada em redes P2P (não utilizando pontos centrais de controle), onde cada instituição participante torna-se um nodo da rede (chamado de BNode) [54].

No BRICKS dados são armazenados em uma DHT, e embora seu principal objetivo esteja relacionado à disponibilidade, a preservação do conteúdo é estatisticamente garantida. Para tal, novas cópias do conteúdo são geradas ou removidas durante a entrada e saída de nodos da rede [36].

Outra característica do projeto BRICKS é a integração com Bibliotecas OAI [54].

Neste sentido, cada BNode comporta-se com um Provedor de Serviços OAI e colhe os metadados, e posteriormente conteúdos, de entidades externas ao sistema. Vale ressaltar que não há interesse, por parte do projeto BRICKS, em trabalhar como parte integrante de uma federação OAI, o objetivo é apenas utilizar o protocolo OAI-PMH para facilitar a importação de conteúdo.

5.2 A abordagem LOCKSS

O sistema LOCKSS (*Lots of Copies Keep Stuff Safe*) é um software *open source* e modela as técnicas que bibliotecas utilizam para preservação de documentos físicos, com o intuito de resolver o problema de acesso e preservação de periódicos (*journals*) em meio digital [41]. Desta forma, uma página aparenta estar sempre disponível – a partir de sua URL original – mesmo que seu editor (publicador) esteja indisponível.

A preservação de acesso aos objetos digitais é realizada através de técnicas comuns, como *links*, *bookmarks* e motores de busca. Para tal, cada uma das bibliotecas deve executar web *caches* (o sistema LOCKSS) que terão como responsabilidade a coleta, distribuição (fornecimento de acesso local) e preservação do conteúdo digital.

5.2.1 Coleta de Conteúdo

A coleta de dados tem como objetivo capturar a versão mais recente de um periódico em específico. Para seu funcionamento, um bibliotecário – administrador do sistema – informa ao LOCKSS o volume do periódico que deseja preservar, através da URL raiz do volume, e a data de freqüência de atualização (ex: mensalmente) [47]. Nesta freqüência, um robô de busca inicia a coleta – a partir da URL raíz – e captura todo conteúdo incluído ou modificado (a captura inclui as sub-árvores de diretório).

A coleta dos dados somente é autorizada se o servidor do publicador verificar que a requisição foi realizada por um IP (*Internet Protocol*) autorizado. Vale ressaltar que a coleta dos dados é totalmente independente do acesso efetuado pelos leitores, ou seja, os *caches* são populados mesmo que o material não seja requisitado pelo leitor.

A Figura 5.1 apresenta um exemplo de coleta de conteúdo pelo sistema LOCKSS.

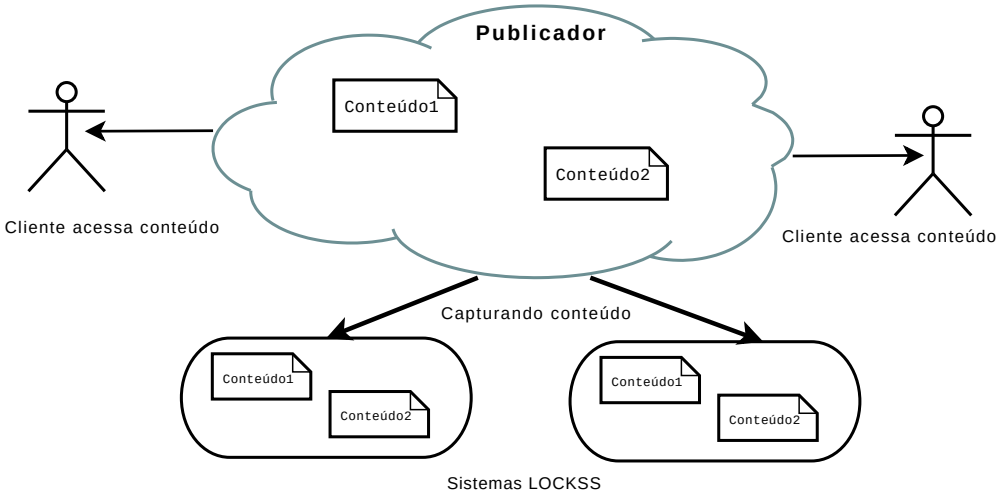

Figura 5.1: Coleta de conteúdo pelo sistema LOCKSS

5.2.2 Preservação de Acesso ao Material

O LOCKSS fornece acesso transparente aos conteúdos por ele preservados, uma vez que intercepta as requisições direcionadas aos periódicos correspondentes. A retenção das requisições é realizada através da integração com um *proxy* web disponível na instituição [11].

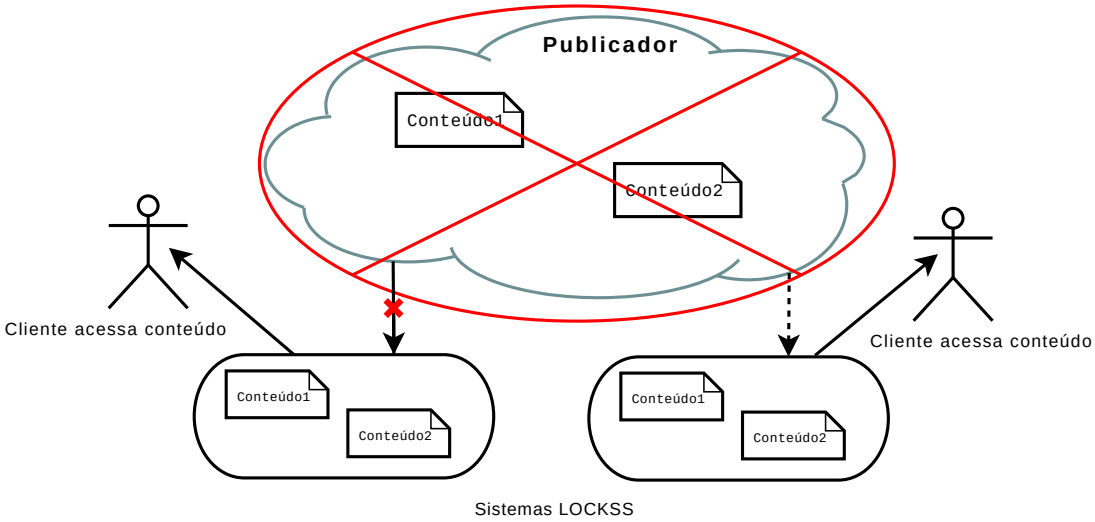

Figura 5.2: Preservação de acesso ao conteúdo fornecido pelo LOCKSS

Quando uma requisição é interceptada, o LOCKSS tenta capturar o conteúdo diretamente do publicador. Se o acesso ao conteúdo requisitado estiver indisponível (ex: servidor

do publicador pode estar inalcançável momentaneamente) uma cópia local é entregue ao requisitante [11, 47]. Um exemplo deste funcionamento pode ser observado na Figura 5.2.

5.2.3 Auditoria e Preservação

A preservação de conteúdo no LOCKSS é realizada através da cooperação entre os *caches* (disponíveis localmente ou através da Internet) que tem por responsabilidade preservar o mesmo conteúdo. Esta cooperação, conforme apresenta a Figura 5.3, tem por intuito detectar e reparar eventuais problemas de integridade no material [47].

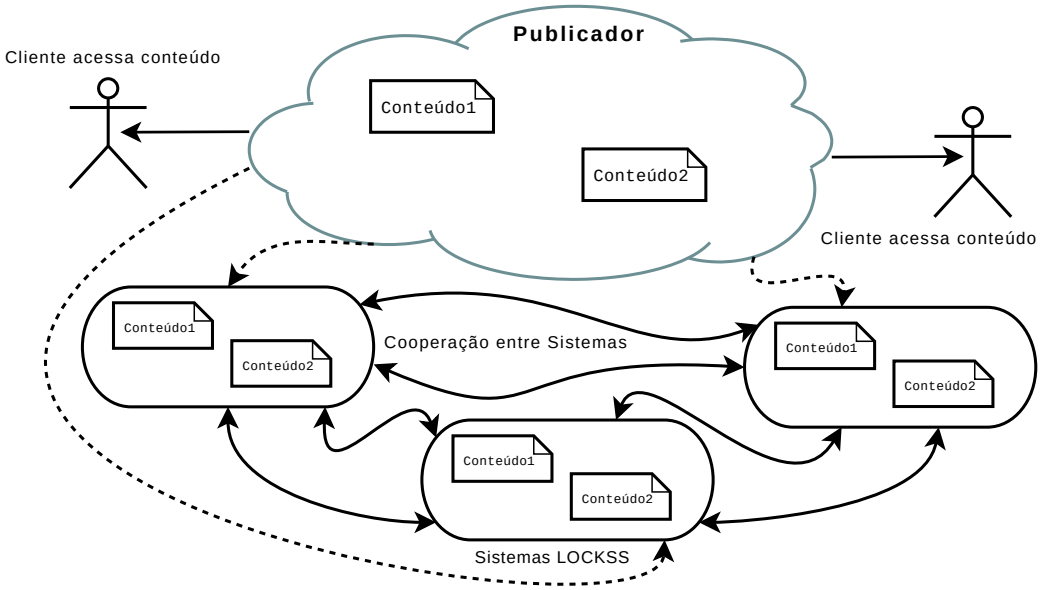

Figura 5.3: Preservação de conteúdo e auditoria no LOCKSS

Neste contexto, os sistemas LOCKSS formam uma rede P2P, organizam e participam de pesquisas de opinião (*options polls*), fornecendo informações ("votos") sobre as condições atuais dos seus objetos digitais (os votos são calculados através de funções *hash*) [46]. Se o conteúdo em um sistema estiver corrompido ou incompleto, este irá "perder" a pesquisa. Sendo assim, seu conteúdo deve ser corrigido baseando-se no material dos sistemas "vencedores".

A cooperação entre os sistemas fornece confiança de que o conteúdo disponível para os leitores estará íntegro quando for requisitado. Além disso, torna-se evidente de que quando mais organizações preservarem o mesmo conteúdo, maior será sua garantia de integridade e disponibilidade [11].

5.3 Considerações Finais

Este Capítulo apresentou duas soluções para preservação digital de longo prazo baseadas em redes P2P. Embora estas soluções sejam amplamente reconhecidas pela comunidade científica, ambas não se integram à federações de Bibliotecas Digitais OAI e consideram que todos os objetos do sistema possuem a mesma importância, do ponto de vista de preservação.

Neste sentido, considera-se que não empregar o protocolo OAI-PMH como principal ferramenta para disseminação de acesso ao conteúdo, acarreta em uma considerável diminuição na visibilidade da informação na Internet, uma vez que este protocolo vem sendo amplamente utilizado e tem se tornado um padrão para o compartilhamento de metadados.

Além disso, não identificar o nível de importância do conteúdo faz com que todos os objetos sejam replicados de maneira idêntica, o que, de forma geral, resulta no mesmo número de cópias para todos os objetos do sistema e impede a otimização da redução de custo com mídias de armazenamento.

Para suprir as necessidades supracitadas, no Capítulo 6 propõe-se uma nova arquitetura. A abordagem em questão considera o protocolo OAI-PMH e a identificação da importância do objeto como fatores chave para a preservação e disseminação de acesso ao conteúdo na Internet.

CAPÍTULO 6

ARQUITETURA OAI PARA PRESERVAÇÃO DIGITAL

Conforme exposto nos Capítulos anteriores, é evidente a necessidade da existência de instrumentos que proporcionem a criação de Bibliotecas Digitais com baixo custo e com alto grau de confiabilidade e disponibilidade.

Neste contexto, mecanismos de preservação dos objetos digitais já são realidade, como pode ser observado nos projetos LOCKSS [46] e BRICKS [54]. No entanto, tais mecanismos não mantêm compatibilidade com o protocolo OAI-PMH e consideram que todos os objetos do sistema possuem a mesma importância, do ponto de vista de preservação.

A proposta deste trabalho é conceber uma arquitetura para preservação digital compatível com o protocolo OAI-PMH e que considere um *fator de confiança* para cada objeto. Entende-se *fator de confiança* como uma medida de probabilidade requerida pelo administrador para se recuperar o objeto em seu formato original (por exemplo, deve ser possível o usuário acessar determinado conteúdo com 98% de certeza).

A idéia central da arquitetura é manter a forma de disseminação de acesso ao conteúdo definida pelo protocolo OAI-PMH e criar um mecanismo, de baixo custo, para preservação dos objetos. A arquitetura proposta baseia-se no pressuposto de que Bibliotecas Digitais OAI possuem o mesmo objetivo, portanto, podem colaborar para criação de um sistema que aumente a confiabilidade e disponibilidade de seus objetos.

As seções subseqüentes apresentam em detalhes a definição da arquitetura e de seus componentes.

6.1 Visão Geral

Conforme apresentado na Seção 2.1, uma confederação de Bibliotecas Digitais OAI é formada à partir de Provedores de Dados e Serviços. Provedores de Dados gerenciam os repositórios de dados e, desta forma são responsáveis pelo armazenamento e fornecimento

dos objetos digitais aos usuários. Além disso, DPs têm como responsabilidade o gerenciamento dos metadados dos objetos, que devem ser criados e fornecidos aos SPs através de uma interface de "colheita".

Embora o esquema proposto pelo protocolo OAI-PMH garanta, de forma eficaz, a disseminação de acesso ao conteúdo na Internet, não faz parte do escopo do protocolo assegurar a preservação do conteúdo digital. Desta forma, objetos podem permanecer inacessíveis ou ser permanentemente perdidos diante de falhas em um único DP, uma vez que o armazenamento do conteúdo é realizado por apenas um DP.

Com o intuito de resolver o problema de preservação digital em confederações de bibliotecas digitais OAI, este trabalho propõe uma arquitetura baseada na colaboração entre os DPs através da formação de uma rede P2P. A utilização de redes P2P formadas por Provedores de Dados OAI tem sido abordada em diversos trabalhos [19, 20, 21, 60]. De forma geral, estas abordagens têm como objetivo aumentar as capacidades de consulta, a partir da criação de métodos de busca distribuída sobre os repositórios de metadados OAI.

Neste trabalho são abordados aspectos relativos à busca distribuída dos objetos digitais, no entanto, o foco principal encontra-se na preservação digital dos objetos, mantendo-os íntegros e disponíveis na rede por longos períodos de tempo.

Assim, a arquitetura posposta define a inserção dos Provedores de Dados OAI em uma Tabela *Hash* Distribuída (DHT) customizada – vide Figura 6.1 – que fornece funções de inserção, busca e auditoria dos objetos. A utilização de uma DHT como base para o sistema de preservação digital em detrimento à redes P2P não estruturadas, dá-se devido sua característica de endereçamento direto aos objetos, o que evita a necessidade de "inundar" a rede (*flooding*) em busca de conteúdo, diminuindo assim o trafego de rede e, conseqüentemente, otimizando os processos de busca.

Como apresentado na Seção 4.2.2, DHTs provêm um mecanismo robusto e escalável para consulta e armazenamento de objetos digitais. Com o intuito de aprimorar os aspectos relativos à sistemas de preservação digital, em nossa arquitetura consideramos um *fator de confiança* (probabilidade desejada de se recuperar um objeto) para cada objeto

digital. Tal fator é fornecido pelo usuário – através da interface de administração do DP – no momento em que o objeto é inserido no sistema, e tem como objetivo identificar os objetos mais importantes do ponto de vista de preservação. Desta forma, objetos com mais importância possuem um maior índice de replicação.

A confiabilidade do objeto é alcançada considerando a *confiabilidade do sítio* (probabilidade de um determinado sítio não falhar) e a replicação do objeto. Conforme exposto no Capítulo 3, para conseguir um alto nível de confiabilidade para um objeto, este é replicado em diferentes nodos da DHT até alcançar o nível de confiabilidade requisitada pelo usuário. Vale ressaltar que a escolha da estratégia de replicação utilizada (uma lista das possíveis estratégias de replicação em protocolos DHT é apresentada em [37]) é diretamente dependente da implementação da DHT. Este trabalho define do ponto de vista arquitetural, o comportamento da DHT e as funções necessárias para garantir da preservação do conteúdo, mas não determina qual estratégia de replicação deve ser utilizada.

Além disso, é importante salientar que não faz parte do escopo deste trabalho definir a maneira utilizada pela DHT para determinar a *confiabilidade do sítio*. No entanto, vale destacar que o cálculo utilizado para obter este parâmetro deve ser definido durante a implementação da DHT, considerando as possíveis falhas listadas na Seção 3.2. Estratégias para estimar a *confiabilidade do sítio* podem ser encontradas em [26].

O acesso à DHT ocorre de forma similar ao utilizado no projeto CAN [45]. Assume-se, desta forma, que a DHT está associada a um domínio DNS (*Domain Name System* – Sistema de Nomes de Domínios [43]), que é resolvido no endereço IP (*Internet Protocol*) de um ou mais *peers* da rede. Uma aplicação externa associada ao DNS, é responsável por manter atualizada uma lista com os *peers* disponíveis na rede. Cabe ao desenvolvedor definir a forma de manter esta lista atualizada, podendo-se utilizar desde técnicas simples (como obter dados de um IP através de *ping*) até técnicas mais complexas (como *group membership* [35]).

É interessante observar que a utilização de um domínio DNS torna o acesso a qualquer nodo do sistema transparente, uma vez que basta ao usuário conhecer a URI (*Uniform*

Resource Identifier) que endereça a DHT. Além disso, a possibilidade de acesso através de múltiplos *peers* distribui o trabalho para atender as requisições dos usuários e aumenta a disponibilidade do sistema.

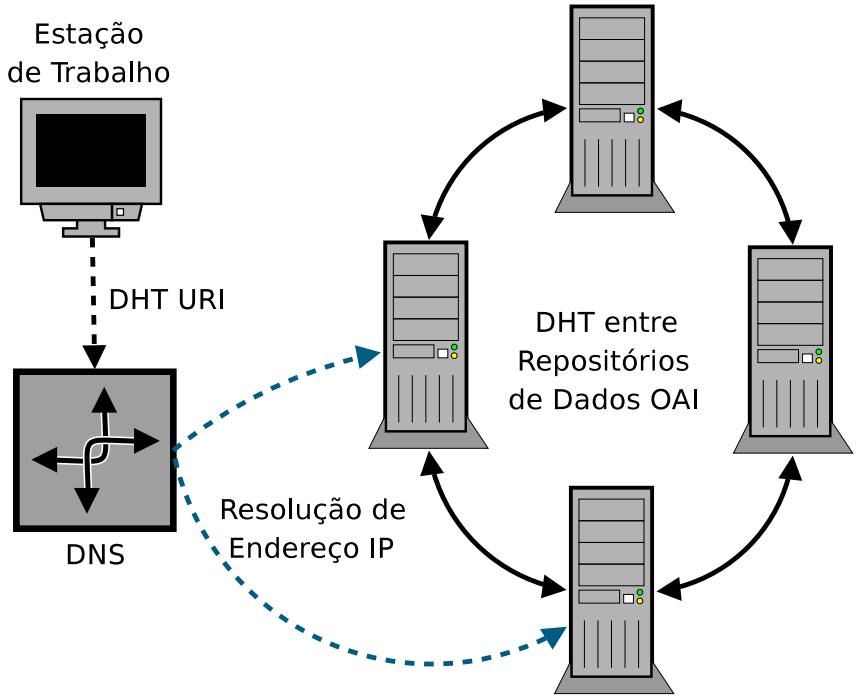

Figura 6.1: Colaboração entre os Repositórios de Dados OAI através de uma DHT. O Acesso à DHT é realizado por de uma única URI, que através de um DNS, é resolvida para qualquer *peer* da rede.

Como pode ser observado, cada nodo do sistema desempenha tanto o papel de Provedor de Dados como de DHT e, desta forma, realiza parte do trabalho de preservação e gerenciamento de metadados. Para manter total compatibilidade com o protocolo OAI-PMH bem como a transparência do sistema, em nível de usuário, os metadados pertencentes a cada um dos *peers* da rede são colhidos por um ou mais Provedores de Serviço, que – com base nos metadados – disponibilizam ao usuário funções complexas de busca.

Neste contexto, como apresentado em detalhes na Seção 6.3.2 e ilustrado na Figura 6.8, para realizar uma busca o usuário acessa um Provedor de Serviços e fornece informações sobre o objeto desejado (título, resumo, autores, etc.). O SP então, realiza uma busca em seus metadados e recupera as informações do objeto, incluindo o DOI (e.g., *http://pdht.c3sl.ufpr.br/1884/7544*), que será utilizado pelo usuário para requisitar

o objeto à DHT.

Ao receber a requisição do usuário, através de qualquer um de seus *peers*, a DHT realiza efetivamente a busca pelo objeto. O parâmetro de consulta utilizado é o identificador do objeto, seguindo os padrões descritos na Seção 2.1. É importante salientar que durante o procedimento de busca, o objeto pode ser encontrado em seu "dono" (*peer* onde ele foi originado) ou em qualquer nodo que possua sua réplica.

Por fim, o usuário acessa o objeto – ou visualiza uma mensagem de erro, quando o objeto não for encontrado – através de uma página *Web* – gerenciada pelo DP – disponível no nodo em que o conteúdo foi localizado.

É importante observar que o DP responde à requisições a partir de uma URI própria, dessa forma, pode ser acessado diretamente (para fins administrativos ou para executar qualquer outra função local por ele disponibilizada). Além e prover uma interface de acesso ao conteúdo para os usuários, o DP é responsável por fornecer as funções necessárias para a "colheita" dos metadados por um SP. Desta forma, ele responde a todos os verbos do protocolo OAI-PMH, conforme apresentado na Seção 2.1.

Além disso, cada DP mantém localmente os metadados dos objetos nos quais ele é "dono". Diferentemente dos objetos (considerados imutáveis), metadados podem sofrer modificações contínuas e, desta forma, precisam ser constantemente colhidos pelos Provedores de Serviço. A coleta periódica dos metadados também pode ser justificada pela inserção de novos objetos na rede.

No contexto apresentado, evidencia-se que os metadados podem ser encontrados em seus respectivos Provedores de Dados ou nos Provedores de Serviço, que por natureza possuem todos os metadados da rede. Desta forma, quando um DP é requisitado a apresentar ao usuário informações sobre um determinado objeto, ele verifica, em primeira instância, se o objeto requisitado o pertence. Se esta premissa for verdadeira, as informações são recuperadas de seus próprios metadados, caso contrário, o metadado é requisitado, em tempo de execução, a um dos provedores de serviço. Embora os metadados possam ser recuperados remotamente, em ambos os casos o acesso ao conteúdo (arquivo) é suprido pelo próprio Provedor de Dados.

6.2 Componentes da Arquitetura

Com o objetivo de fornecer informações detalhadas sobre cada um dos componentes da arquitetura, a Subseção 6.2.1 descreve as funcionalidades da DHT e a Subseção 6.2.2 apresenta em detalhes como um Provedor de Dados é estruturado. Considerações sobre o funcionamento do Provedor de Serviços são apresentadas na Subseção 6.2.3.

6.2.1 DHT Confiável

A DHT é o componente da arquitetura responsável por realizar efetivamente a preservação digital do conteúdo. Seu papel é garantir a confiabilidade dos objetos digitais por longos períodos de tempo (anos ou décadas). Para tal, ela realiza as seguintes tarefas:

1. Aceita requisições para a inserção de objetos digitais na rede, considerando um *fator de confiança* – definido pelo administrador – para cada objeto digital;

2. Calcula quantas réplicas do objeto são necessárias para garantir a confiabilidade requisitada pelo usuário;

3. Executa a replicação dos objetos digitais; e

4. Realiza auditoria de cada objeto através da comparação das várias réplicas e, se necessário, corrige as cópias corrompidas.

Além de garantir a preservação dos objetos, a utilização da DHT aumenta a disponibilidade do conteúdo, uma vez que os objetos são replicados e podem ser recuperados de diferentes pontos do sistema. Mais do que isso, a utilização de um DNS para endereçar a DHT, conforme exposto na Seção 6, aumenta a disponibilidade do sistema como um todo, já que requisições podem ser atendidas por qualquer nodo da rede, o que evita um ponto central de falha.

Para realizar as tarefas listadas anteriormente, é necessária a customização dos algoritmos de inserção e busca, além da concepção de mecanismos de auditoria. Desta forma,

assume-se que o sistema de preservação digital utilizado nesta arquitetura deve ser, preferencialmente, construído sobre algum dos protocolos DHT existentes ([45, 49, 53]) e desta forma, utilizar-se de soluções já consolidadas e aceitas pela comunidade acadêmica.

Além disso, é importante enfatizar que o principal objetivo da arquitetura proposta é a preservação do conteúdo digital, em seu formato original, por longos períodos de tempo. Sendo assim, objetos são considerados imutáveis (como proposto em [39]), o que torna desnecessário a existência de um algoritmo para atualização do conteúdo.

Neste sentido, as próximas Seções descrevem, de forma abstrata, o funcionamento de cada um dos algoritmos necessários para a preservação digital do conteúdo na DHT.

6.2.1.1 Algoritmo de Inserção na DHT

O objetivo do algoritmo de inserção é armazenar um objeto digital na rede satisfazendo a confiabilidade desejada. A *confiabilidade do sítio* e o *fator de confiança* do objeto, fornecido pelo administrador, são a base para descobrir a quantidade de réplicas necessárias para alcançar a confiabilidade desejada para o conteúdo no sistema.

Embora a arquitetura proposta não determine qual estratégia de replicação deve ser utilizada pela DHT (deixando a escolha livre ao desenvolvedor), é estritamente recomendado que a estratégia escolhida possa ser aplicada sobre os protocolos DHT existentes, conforme exposto na Seção anterior.

Além disso, uma vez que a arquitetura proposta utiliza-se de um *fator de confiança* para cada objeto, se faz necessário a escolha de estratégias de replicação flexíveis, isto é, que permitam estabelecer "políticas" de replicação diferenciadas para cada objeto. Pode-se considerar tal afirmação pertinente, já que algumas estratégias de replicação (como por exemplo a replicação simétrica [30]) exigem o mesmo número de réplicas para todos objetos na rede, o que inviabiliza a solução proposta neste trabalho.

Considerando o contexto apresentado, o algoritmo de inserção deve possuir a seguinte interface:

`insert(name, value, reliability)`: Quando um objeto é inserido na rede, o administrador informa o nome, o conteúdo e o *fator de confiança*, neste caso definidos

por `name`, `value` e `reliability`, respectivamente. Posteriormente, com base na estratégia de replicação adotada, o objeto é mapeado em diferentes nodos da DHT, através do operador padrão `put(key, value)`, até satisfazer a confiabilidade exigida.

A Figura 6.2 ilustra o fluxo a ser seguido pelo algoritmo durante a inserção de um objeto na DHT.

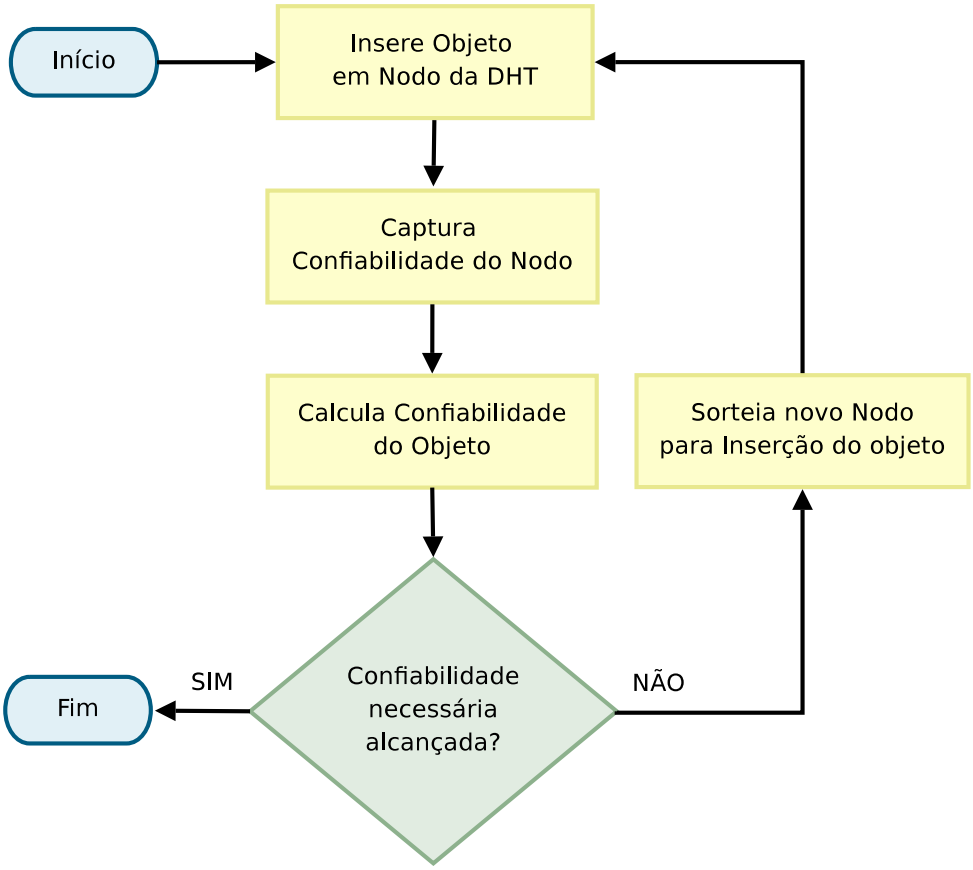

Figura 6.2: Fluxograma do algoritmo de inserção da DHT

Com base no fluxo ilustrado na Figura 6.2 é importante considerar que, inicialmente o conteúdo é inserido localmente, ou seja, no sítio que recebeu a requisição pela inserção. Além disso, a seleção de novos nodos (para replicar o objeto) dá-se através da estratégia de replicação escolhida pelo projetista/desenvolvedor.

Também é importante observar que objetos inseridos são armazenados fisicamente em um sistema de arquivos – acessado posteriormente pelo DP para fornecer o objeto ao usuário – e seus nomes devem respeitar o padrão, descrito na Seção 2.1, para identificação de objetos (DOI).

Desta forma, ao desmembrar o identificador *pdht.c3sl.ufpr.br/1884/7544*, os parâmetros **1884** e **7544** tornam-se diretórios, sendo que o segundo é inserido dentro do primeiro, respeitando a hierarquia da própria URI. Além disso, o arquivo armazenado em disco é nomeado com o valor do segundo parâmetro (**7544**) e é inserido dentro do diretório com o mesmo nome.

Esta estruturação faz com que conteúdos armazenados em disco possam ser localizados pelos DPs utilizando-se somente do DOI, o que torna desnecessário a existência de informações adicionais para a localização do objeto no sistema.

Além disso, a utilização do DOI como nome do objeto pela DHT, garante a compatibilidade com o protocolo OAI-PMH, uma vez que permite que buscas sejam realizadas diretamente pelo identificador do objeto, obtido a partir de um dos Provedores de Serviço ou, eventualmente, de mecanismos de busca na Internet. Internamente, considerando a estratégia de replicação escolhida, a DHT deve utilizar o nome do objeto para geração de uma ou mais *hashs* que indexam o conteúdo.

6.2.1.2 Algoritmo de Busca na DHT

O algoritmo de busca tem por objetivo encontrar o objeto solicitado – a partir de seu identificador (DOI) – de forma distribuída, considerando todas as réplicas do sistema. A busca por um objeto pode ser iniciada por qualquer nodo da DHT e tanto o conteúdo original como uma réplica do objeto podem ser retornadas. Cabe ao DNS, conforme exposto na Seção 6.1, decidir o sítio responsável por receber cada uma das requisições.

O algoritmo deve possuir a interface `lookup(name)` e para endereçar os objetos utilizar o operador padrão `get(key)`, considerando a estratégia de replicação adotada. A Figura 6.3 ilustra o fluxo a ser seguido pelo algoritmo.

Como pode ser observador no fluxo apresentado na Figura 6.3, após realizar a busca a DHT não fornece o objeto diretamente ao usuário. O acesso ao conteúdo é realizado a partir de uma página *Web* – gerenciada pelo DP – disponível no sítio em que o conteúdo foi encontrado. Este passo é necessário para que o usuário possa acessar os detalhes do objeto (informações contidas nos metadados), em conformidade com o protocolo OAI-PMH, antes

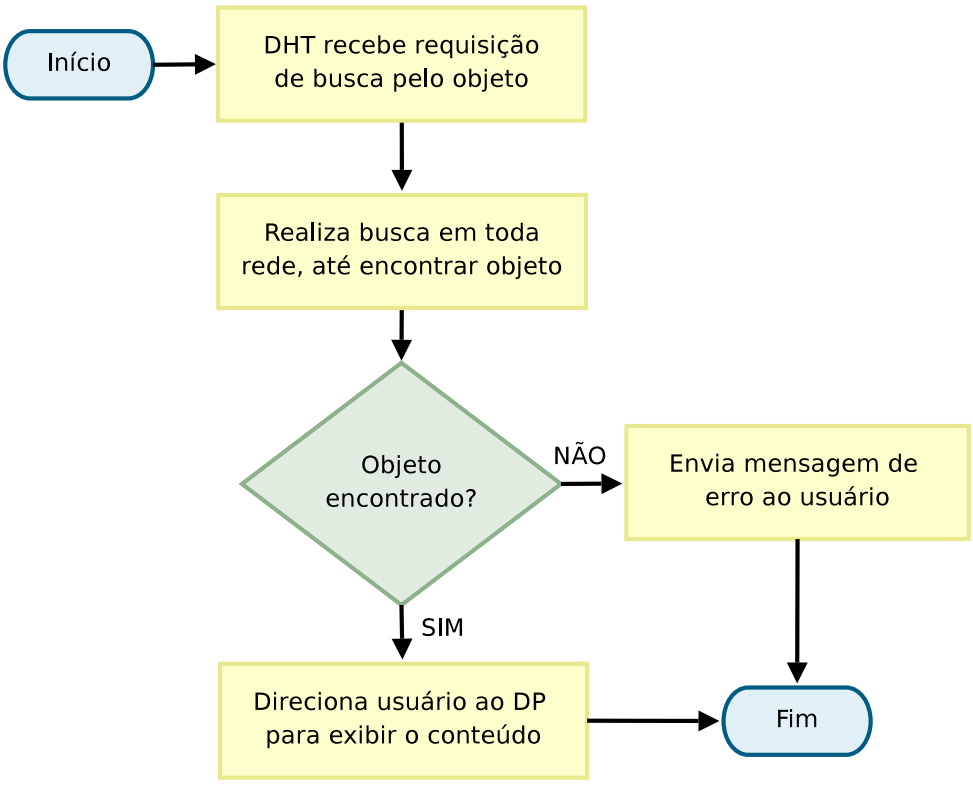

Figura 6.3: Fluxograma do algoritmo de busca da DHT

de efetuar efetivamente o *download* do conteúdo. Se o objeto não for encontrado, uma mensagem padrão deve ser apresentada ao usuário, através do DP, informando que o objeto se encontra indisponível no sistema.

Requisições são recebidas pelo Provedor de Dados através de uma URI própria (e.g., *dp.c3sl.ufpr.br*), informada à DHT no momento da configuração/instalação do sistema. Além disso, para que o objeto possa ser identificado pelo Provedor da Dados, parâmetros de identificação do DP e do objeto são enviados juntamente com a URI, conforme ocorre na atual versão do protocolo OAI-PMH. Nota-se desta forma, que a DHT recebe uma requisição através da URI *pdht.c3sl.ufpr.br/1884/7544* e como resposta retorna a URI *dp.c3sl.ufpr.br/1884/7544*, identificando um dos DPs que armazenam uma cópia do objeto.

É importante observar que, desta maneira, além de realizar seu papel de preservação, a DHT pode suprir as funcionalidades desempenhadas pelo *Handler Server* (HS), que traduz a URI acessada pelo usuário para a URI do Provedor de Dados. Para tal, basta alterar no metadado o endereço do HS (e.g., *hdl.handle.net*) para o endereço da DHT

(e.g., *pdht.c3sl.ufpr.br*).

6.2.1.3 Algoritmo de Auditoria

Como apresentado no Capítulo 3, para que a preservação do conteúdo em sistemas de preservação digital seja garantida, deve-se assegurar integridade e autenticidade dos objetos. Neste sentido, mecanismos de auditoria devem ser concebidos, para garantir que a possibilidade de acessar objetos corrompidos no sistema, por parte do usuário, seja mínima.

Sendo assim, na arquitetura proposta, além da DHT realizar as operações de inserção, replicação e busca distribuída dos objetos digitais, é sua responsabilidade auditar a integridade dos objetos. Para tal, o algoritmo de auditoria proposto tem por objetivo detectar e corrigir modificações inesperadas no conteúdo dos objetos, mantendo-os íntegros e, conseqüentemente, tornando-os mais confiáveis.

A estratégia de auditoria adotada é similar a utilizada pelo projeto LOCKSS, que periodicamente audita objetos específicos. Na primeira etapa do algoritmo, uma *hash*, que sumariza o conteúdo do objeto, é gerada para cada uma das cópias do objeto auditado. No término da etapa I, o sistema gera uma *hash* confiável h, que representa o maior número de cópias idênticas do objeto. Por fim, o algoritmo compara h com todas as *hashs* geradas, com o intuito de identificar divergências. Assim, se a *hash* de um objeto for diferente de h, este objeto é considerado corrompido e é restaurado pelo sistema.

Internamente a DHT deve possuir mecanismos que identifiquem objetos recentemente auditados, e selecionar de forma aleatória, novos objetos que serão analisados pelo processo de auditoria, considerando sua última verificação. A periodicidade do processo de auditoria deve ser previamente definida pelo projetista/desenvolvedor e, possivelmente, alterada pelo administrador do sistema através de um instrumento de configuração.

O algoritmo deve possuir a interface `audit(name)`, onde `name` representa o identificador do objeto (DOI). A Figura 6.4 apresenta, em linhas gerais, os passos a serem seguidos durante a execução do algoritmo.

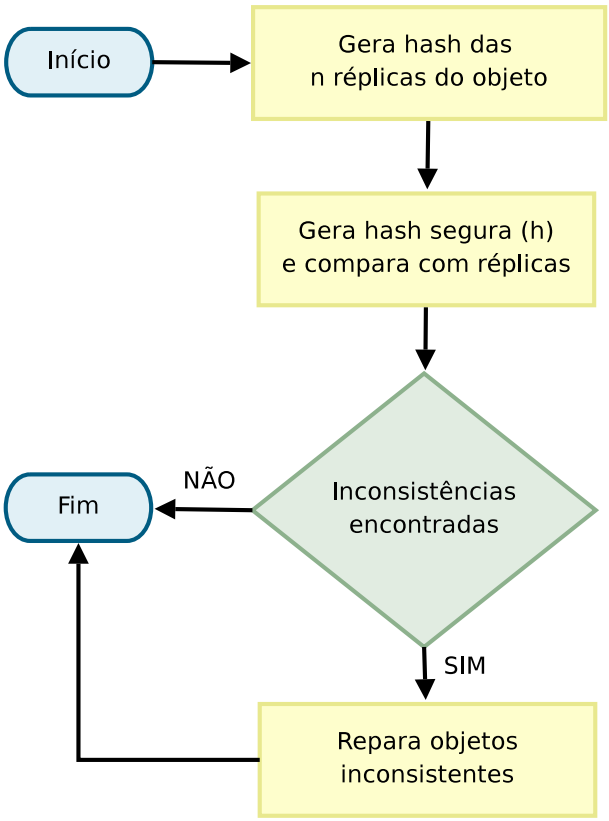

Figura 6.4: Fluxograma do algoritmo de auditoria da DHT

Vale destacar que o algoritmo de auditoria – como os demais apresentados – aproveita-se da topologia da rede P2P para gerar cooperação entre os sítios e, conseqüentemente, garantir a integridade do conteúdo. Além disso, é importante deixar claro que a integridade e disponibilidade dos objetos estão diretamente relacionadas à quantidade de sítios que o preservam. Como apresentado no Capítulo 3, quanto maior o número de réplicas, maior será a garantia de recuperar o objeto da rede em sua forma original.

6.2.2 Provedor de Dados

Conforme exposto anteriormente, cada nodo do sistema é composto por uma DHT e um Provedor de Dados, que de maneira geral tem por objetivo gerenciar o repositório de dados e expor os metadados dos objetos aos Provedores de Serviço (*harvesters*). Provedores de Dados também devem ser capazes de processar requisições baseadas nos verbos do protocolo OAI-PMH, como descrito na Seção 2.1.

Além de se preocupar com seu conjunto básico de requisitos, em nossa arquitetura

DPs devem estar aptos a atender quesitos específicos, como capturar o *fator de confiança* do objeto e realizar comunicação direta com a DHT e Provedores de Serviços, a fim de garantir a preservação e acesso ao conteúdo. Uma vez que diversos *softwares* neste sentido já foram concebidos – como DSpace [2] ou Eprints [5] – é recomendado a utilização e adaptação de tais *softwares* a fim de facilitar o desenvolvimento e garantir os requisitos estabelecidos pela arquitetura.

A próxima Seção apresenta o conjunto mínimo de componentes necessários para o funcionamento de um Provedor de Dados no contexto apresentado.

6.2.2.1 Componentes básicos de um Provedor de Dados

Para facilitar o entendimento, a Figura 6.5 apresenta a estrutura de um DP dividida em camadas.

A *camada de aplicação* é responsável por disponibilizar as interfaces necessárias para comunicação do DP com Provedores de Serviço, bem como fornecer acesso aos usuários e administradores do sistema. Esta camada é composta por três componentes básicos:

- **Interface *Web***: Exerce duas funções principais, (i) fornece acesso ao administrador para realizar inserção de objetos; e (ii) permite o acesso ao conteúdo por parte dos usuários;

- **Processador de Requisições OAI**: Responsável por processar os 6 verbos disponíveis no protocolo OAI-PMH [24], expondo assim os metadados dos objetos aos Provedores de Serviço;

- ***Harvester* de Metadados**: Responsável por recuperar metadados de objetos do Provedor de Serviços. Tendo em vista o grande custo para replicar os metadados dos objetos – já que estes são passíveis de modificações – este componente é de suma importância, uma vez que permite que metadados de objetos replicados sejam recuperados diretamente do SP, que, por natureza, armazena todos os metadados do sistema.

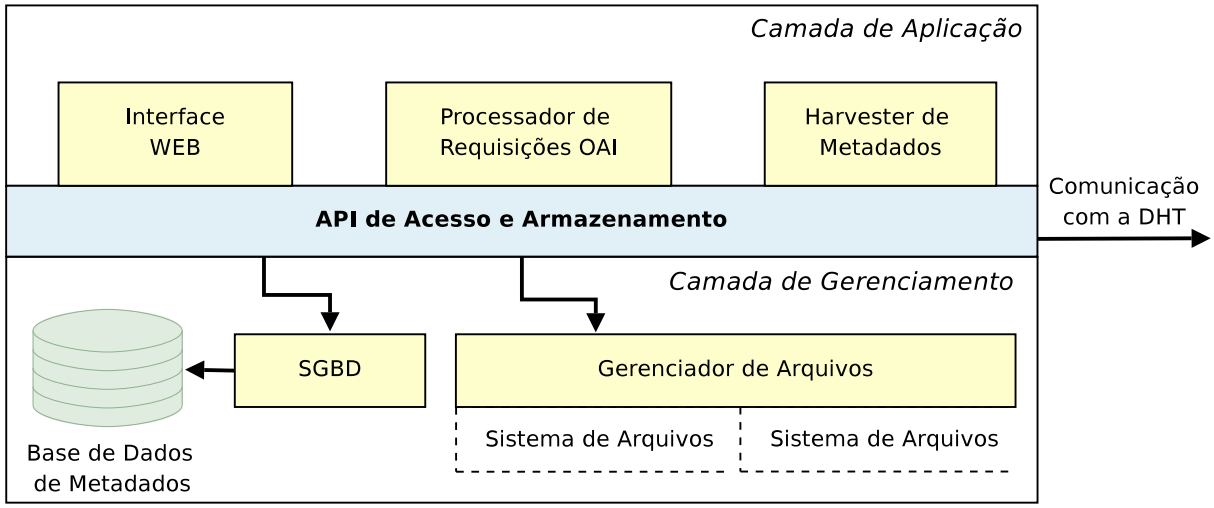

Figura 6.5: Componentes básicos de um Provedor de Dados divididos em Camadas

A *camada de gerenciamento* é responsável gerir os metadados e o conteúdo dos objetos no sistema. Esta camada é composta por um Sistema Gerenciador de Banco de Dados (SGBD) ou qualquer outro mecanismo responsável por armazenar e gerenciar os metadados dos objetos digitais; e um gerenciador de arquivos, responsável por criar um *link* para acesso ao objeto – baseado no padrão de nomes de objetos definido na Seção 6.2.1.1 – durante o processo de busca, e mapeá-lo em uma página *Web* acessível aos usuários.

A comunicação entre as camadas de aplicação e gerenciamento é realizada através da API (*Application Programming Interface*) de acesso e armazenamento, que também fornece os métodos necessários para comunicação entre o DP e a DHT.

Além de sua estruturação, é importante observar que DPs, quando inseridos em uma confederação de bibliotecas OAI, possuem características peculiares. Neste sentido, DPs são devidamente registrados na Iniciativa *Open Archives* e possuem um identificador único, utilizado para compor o identificador do objeto. Tendo em vista que identificadores internos dos objetos (que também compõem o DOI) são gerados de forma independente por cada um dos DPs – e podem se repetir – o identificador do DP garante que objetos de diferentes repositórios sejam identificados de forma distinta e recuperados sem existência de conflitos.

6.2.3 Provedor de Serviços

Provedores de Serviço são responsáveis por realizar a colheita dos metadados em conformidade com os verbos existentes no protocolo OAI-PMH. Como exposto na Seção 2.1, colheitas podem ser baseadas em dados ou em conjuntos, possibilitando a existência de SPs que enderecem apenas conteúdos de assuntos específicos.

Do ponto de vista da arquitetura proposta, é definido a existência de SPs que realizam colheita de todos os metadados do sistema. Esta definição é necessária, pois, no contexto apresentado, além de atuarem como pontos iniciais de busca de conteúdo – por parte dos usuários – alguns SPs desempenham o papel de "fornecedores" de metadados para os Provedores de Dados. Uma lista contendo o endereço dos SPs fornecedores deve ser mantida pelo Provedor de Dados.

Para realizar o fornecimento de metadados aos DPs, Provedores de Serviço aceitam requisições através do verbo `GetRecord`. Como ocorre no DPs, este verbo recebe por parâmetro o DOI e retorna o metadado correspondente. Requisições são realizadas pelos DPs (através do componente Harvester de metadados) via protocolo HTTP (*Hypertext Transfer Protocol*), e os metadados entregues seguindo o padrão *Dublin Core* [3]. A Figura 6.6 ilustra a troca de mensagens entre os provedores de dados e serviços.

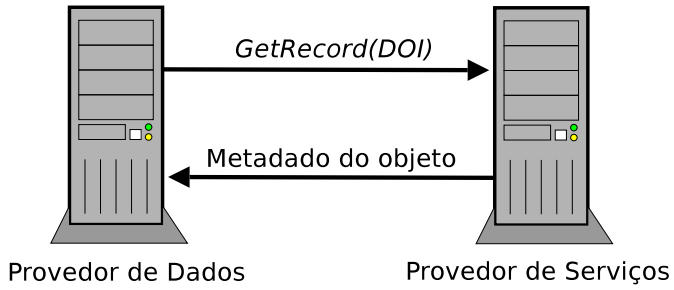

Figura 6.6: Provedor de Dados solicita metadado ao Provedor de Serviço através do verbo `GetRecord`

É importante observar que se o metadado não for encontrado, o SP deve retornar ao DP uma mensagem em formato XML contendo a *tag* `error`, seguindo os padrões de erro descritos para o verbo `GetRecord` em [24].

Além disso, pode-se dizer que a possibilidade de acesso ao verbo `GetRecord` no SP acrescenta ao protocolo OAI-PMH a característica de bilateralidade, ou seja, a existência

de comunicação direta entre os Provedores de Dados e Serviços de forma independente. No entanto, é importante deixar claro que este é um requisito inerente à nossa arquitetura, e conserva, em sua totalidade, a arquitetura do protocolo OAI-PMH descrita em [24].

6.3 Colaboração entre os Componentes

Uma vez que todos os componentes da arquitetura foram apresentados em detalhes, torna-se pertinente minudenciar o fluxo de mensagens entre os componentes para realizar operações básicas de inserção e pesquisa. Desta forma, as seções subseqüentes apresentam em detalhes a comunicação interna entre os componentes da arquitetura.

6.3.1 Inserção de um objeto no Sistema

Conforme pode ser observado na Figura 6.7, durante o processo de inserção de um objeto na rede é necessário o preenchimento de atributos específicos, utilizados durante a criação de seu respectivo metadado.

Desta forma, na primeira etapa do processo, o administrador fornece ao sistema informações descritivas do objeto (como título, autores, resumo, etc.) e realiza o *upload* do conteúdo (arquivo), através de uma interface *Web* disponível no DP. Ainda nesta etapa é fornecido ao sistema o *fator de confiança* do objeto (por exemplo, 95%), que será utilizado como parâmetro para descobrir o número de réplicas necessárias no sistema.

Durante a segunda etapa um código interno do objeto é gerado pelo DP, através do componente SGDB. O código em questão identifica internamente o objeto no DP e é utilizado para compor o identificador do objeto no sistema (DOI). Assim, com base no DOI, recém criado, e nas informações fornecidas pelo administrador na etapa anterior é gerado – e disponibilizado aos SPs – o metadado do objeto digital.

Na terceira etapa do processo o DP envia à DHT uma requisição para inserção do objeto. Tal requisição é enviada ao nodo local da DHT (nodo situado no mesmo repositório que o DP) juntamente com todas as informações fornecidas pelo administrador.

Por fim, na etapa quatro o objeto digital é inserido na DHT, através do método

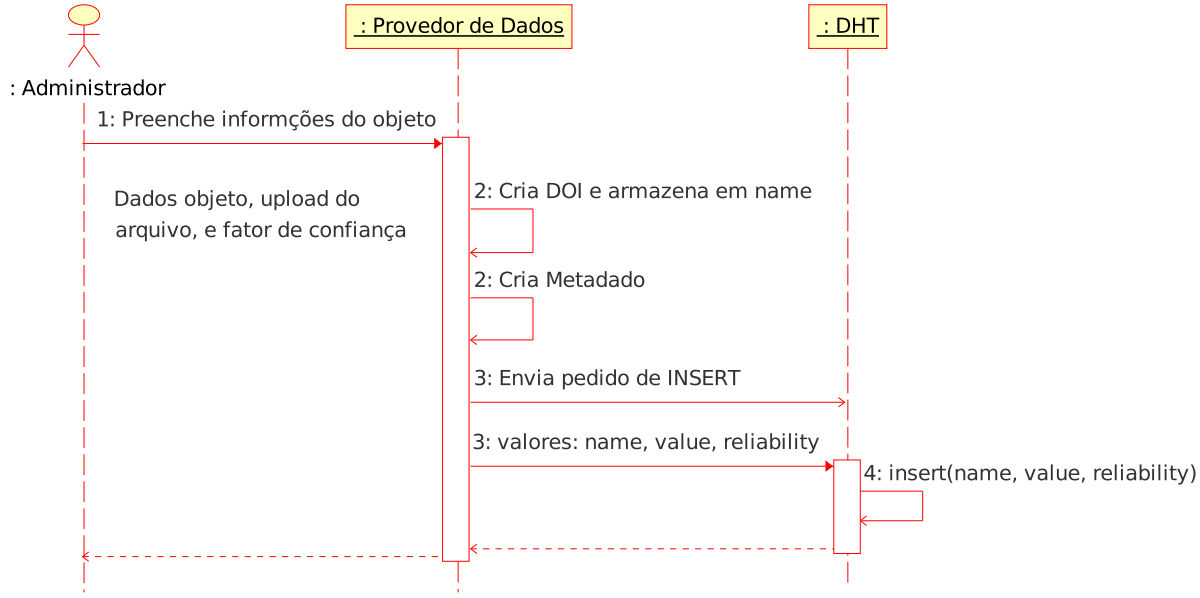

Figura 6.7: Troca de mensagens durante a inserção de um objeto na rede a partir do Provedor de Dados

`insert(name, value, reliability)`, respeitando o curso descrito na Seção 6.2.1.1. Neste instante, os atributos `name`, `value` e `reliability` são preenchidos com o DOI, o conteúdo do arquivo e o *fator de confiança* do objeto, respectivamente. Desta forma, a DHT se responsabiliza pela replicação do conteúdo na rede.

6.3.2 Busca de um objeto no Sistema

A busca de um objeto é iniciada pelo usuário a partir do acesso a um Provedor de Serviços, onde é possível descobrir – através dos metadados colhidos – a URI que identifica o objeto no sistema. A URI inserida no metadado pode endereçar a DHT ou o HS, que por sua vez direciona o fluxo para a DHT.

Neste sentido, é de suma importância observar que o papel de "proxy" do HS na arquitetura pode ser suprido pela própria DHT, uma vez que não é necessário acessar o HS durante a requisição de acesso a um objeto no sistema. Esta característica aumenta, de forma considerável, a disponibilidade do sistema, já que na atual versão do protocolo OAI-PMH o HS pode ser considerado um ponto central de gargalo e falha.

Tendo em vista que a identificação do objeto foi recuperada, através da URI, é possível solicitar a busca do objeto à DHT. É importante salientar que, conforme exposto nas

seções anteriores, é possível recuperar o objeto encaminhando a requisição para qualquer sítio pertencente a DHT. Cabe ao DNS decidir qual sítio será responsável por receber a requisição.

Logo que um pedido de busca é recebido pela DHT, o método `lookup(name)` é invocado, conforme ilustra a Figura 6.8, realizando assim a busca do objeto na rede. Como resposta, a DHT retorna ao usuário uma URI que endereça seu respectivo DP (DP situado no mesmo sítio). Assim, o usuário é automaticamente direcionado para o DP (etapa 6), onde é possível visualizar os detalhes do objeto, ou uma mensagem de erro.

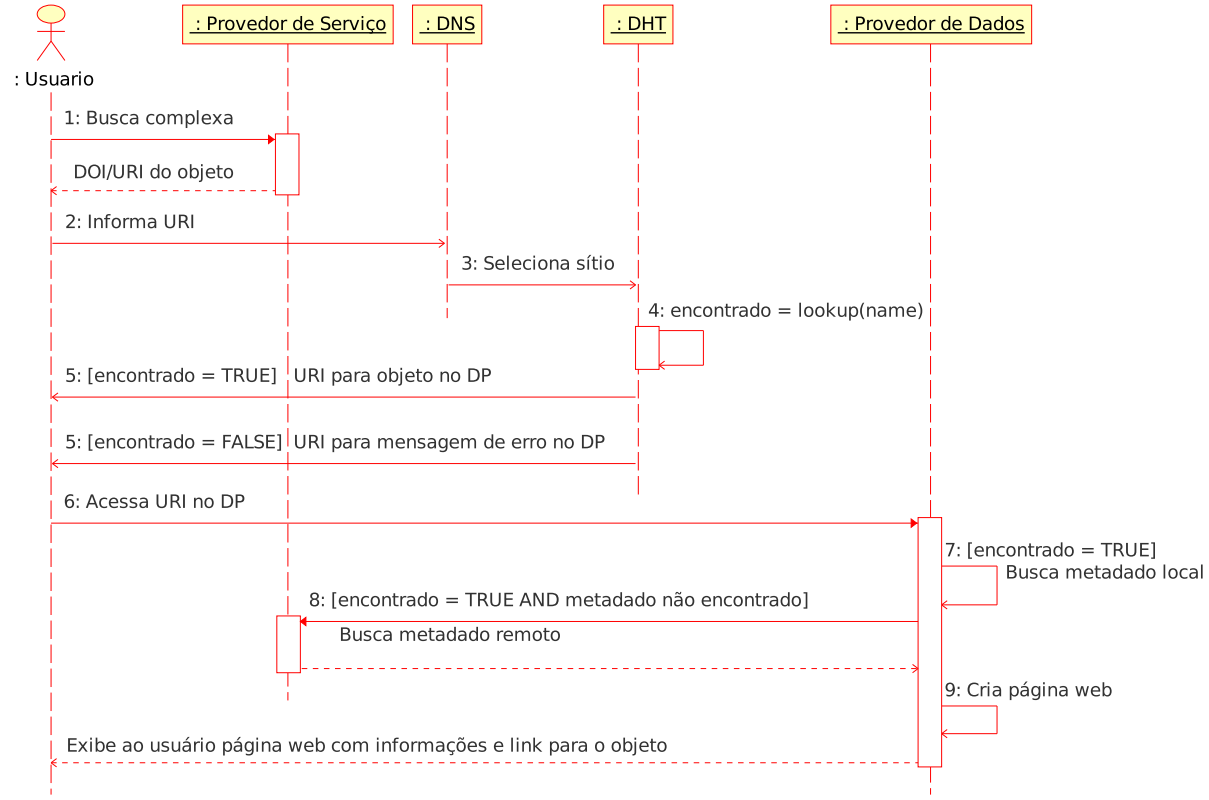

Figura 6.8: Troca de mensagens durante a busca de um objeto na rede

Tendo em vista que as informações descritivas do objeto encontram-se em seu respectivo metadado, é necessário o acesso ao metadado por parte do Provedor de Dados, para que os detalhes do objeto solicitado, bem como o *link* para *download* do conteúdo, sejam apresentados ao usuário. O DP pode obter acesso ao metadado de duas diferentes maneiras:

- **Acesso local**: Uma vez que os DPs armazenam localmente metadados dos objetos

os quais são "donos", inicialmente o DP tenta recuperar o metadado de sua base local, acessando diretamente o componente SGBD. Caso o metadado não seja encontrado, é necessário a utilização do acesso remoto para que a informação possa ser recuperada.

- **Acesso remoto**: O DP faz uma requisição ao SP utilizando-se do verbo *GetRecord*, que a partir de um identificador, retorna o metadado do objeto correspondente.

Por fim, uma página *Web* contendo as informações do objeto, ou uma mensagem de erro, é criada e apresentada ao usuário.

6.4 Considerações Finais

A arquitetura proposta neste trabalho possibilita a integração de Provedores de Dados com o intuito de formar uma rede de preservação digital para garantir confiabilidade e disponibilidade de conteúdo. Se comparada a outros trabalhos, evidenciam-se, de forma clara, duas principais vantagens desta arquitetura.

A primeira refere-se à integração dos sistemas construídos com base na arquitetura com federações de Bibliotecas Digitais OAI, através da utilização do protocolo OAI-PMH. Tal integração é de suma importância, tendo em vista que este protocolo é amplamente utilizado e pode ser considerado um padrão no que diz respeito à disseminação de acesso ao conteúdo na Internet.

A segunda vantagem é a associação de um *fator de confiança* para cada objeto digital, o que permite o administrador informar ao sistema quais objetos devem ser considerados mais importantes do ponto de vista de preservação. Esta característica possibilita a customização da preservação dos objetos, reduzindo o número de réplicas desnecessárias no sistema e otimizando o trafego rede. Por conseqüência, esta solução conduz para redução do custo do sistema, principalmente no que diz respeito à mídias de armazenamento.

Além das vantagens supracitadas, é possível constatar a relação entre o modelo OAIS (vide Seção 3.1) e a arquitetura proposta. Neste sentido, nota-se a presença do Provedor de Dados nas entidades funcionais *Produção*, *Acesso* e *Administração*, enquanto a DHT

relaciona-se às entidades *Gerenciamento de Dados*, *Plano de Preservação* e *Armazenamento*.

Por fim, é importante observar algumas diferenças entre a solução proposta e as abordagens existentes. Enquanto o sistema LOCKSS realiza atualização do conteúdo durante seu processo de coleta, nossa arquitetura preocupa-se em manter os objetos imutáveis e permanentemente preservados, como proposto em [39]. Outro aspecto importante é que no BRICKS, embora a estratégia de replicação utilizada garanta – em nível de *bits* – a preservação do conteúdo, não existe diferenciação entre a importância dos objetos na rede. Desta forma, conteúdos menos importantes podem sobrecarregar a rede, impossibilitando melhores níveis de preservação para conteúdos essenciais.

Com o intuito de validar o funcionamento da arquitetura proposta, o Capítulo 7 apresenta os resultados experimentais obtidos através da implementação de protótipos não funcionais para os componentes da arquitetura.

CAPÍTULO 7

EXPERIMENTOS

Bibliotecas Digitais disponibilizam coleções de conteúdos em meio digital de forma *on-line*, possibilitando acesso a materiais com agilidade, independente de sua localidade.

Conforme apresentado anteriormente, devido a importância destas entidades, torna-se evidente a necessidade da preservação de conteúdo em Bibliotecas Digitais. Neste sentido, uma proposta de arquitetura para preservação digital sobre Bibliotecas Digitais OAI foi descrita no Capítulo 6. A arquitetura especifica as funções, comportamentos e a comunicação entre quatro principais componentes:

- Provedor de Dados;
- Provedor de Serviços;
- Tabela *Hash* Distribuída – DHT; e
- Sistema de Nomes de Domínios – DNS.

Cada um dos componentes apresentados possui atribuições e características distintas, no que diz respeito à preservação e acesso ao conteúdo pelos usuários. Desta forma, neste capítulo conceitos anteriormente apresentados serão abordados de forma prática.

De maneira geral, os experimentos consistem na criação de protótipos não funcionais[1], que possibilitem a validação dos fluxos de mensagens durante o processo de inserção e busca de objetos no sistema. Neste sentido, foram implementados protótipos para os quatro componentes supracitados, bem como para um quinto componente (chamado de navegador), que inicia as requisições e "simula" a ação de um nevegador *Web*.

Sendo assim, as próximas seções descrevem o funcionamento dos protótipos e os testes realizados para validar os fluxos de mensagens descritos no Capítulo 6.

[1]Entende-se **protótipo não funcional** como um "sistema" que não se assemelha ao produto final. Desta forma, este tipo de protótipo é útil para a exploração e testes na fase inicial de desenvolvimento.

7.1 Tecnologias Utilizadas

Antes de abordar aspectos pertinentes à implementação dos protótipos é conveniente apresentar, de forma breve, as tecnologias utilizadas durante o processo de desenvolvimento.

Linguagem de Programação Java: Java é uma linguagem de programação desenvolvida pela *Sun Microsystems* em 1995. Sua sintaxe básica é derivada de linguagens como C e C++, e tem por característica a facilidade no desenvolvimento orientado à objetos. De maneira geral, aplicações Java são compiladas (gerando um código intermediário) e posteriormente interpretadas por uma máquina virtual (JVM), possibilitando o funcionamento em diferentes sistemas operacionais e plataformas computacionais.

Internet *Sockets*: De maneira geral, um Internet *Socket* [59] (comumente conhecido como *Socket* de rede ou somente *Socket*) é o ponto final de um fluxo de comunicação bidirecional em redes IP. Cada *Socket* é mapeado em uma aplicação, e fornece uma interface entre as aplicações e a pilha de protocolos TCP/IP.

***eXtensive Markup Language* - XML:** O *eXtensible Markup Language* (XML) é uma linguagem de marcação de texto – desenvolvida pelo *World Wide Web Consortium* (W3C) – flexível, extensiva e auto-descritiva, que permite a criação de padrões com elementos e atributos pré-definidos. Devido sua simplicidade, o XML é utilizado em diversos tipos de contexto, principalmente no que diz respeito à troca de dados entre aplicações [6].

Considerando as tecnologias apresentadas, vale destacar que a linguagem Java (mais especificamente Java 6 *OpenJDK*) foi utilizada durante a implementação de todos os componentes (protótipos) utilizados nos experimentos.

A comunicação entre os componentes dá-se através do uso de Internet *Sockets*. Além disso, é importante considerar que as trocas de mensagens são orientadas à conexão (ocorrem sobre os protocolos TCP/IP), a fim de facilitar a identificação de falhas.

Por fim, os testes de validação foram realizados sobre o sistema operacional GNU/Linux, utilizando-se as distribuições Debian 4.0 e Ubuntu 8.04.

7.2 Descrição dos Componentes

As seções subseqüentes descrevem o comportamento e as funções realizadas por cada um dos componentes propostos pela arquitetura. Finalizando, as Seções 7.3.1 e 7.3.2 apresentam os testes realizados para validar os fluxos de inserção e busca de objetos, respectivamente.

7.2.1 Componente Navegador

Conforme mencionado anteriormente, o componente navegador simula o funcionamento de um *Web browser*, ou seja, é o cliente responsável por prover uma interface entre o usuário e o sistema de preservação digital. Este componente possui quatro principais atribuições:

1. **Requisitar inserções de objetos**: Com base nas informações fornecidas pelo administrador, o componente faz uma requisição para que o DP realize a inserção de um objeto;

2. **Requisitar buscas complexas por objetos**: A partir do título do objeto, o componente faz uma requisição a um SP e recebe como resposta o identificador do objeto (DOI);

3. **Solicitar buscas pelo conteúdo no sistema**: Utilizando-se do DOI recuperado na etapa anterior – ou possivelmente de mecanismos de busca na Internet – o componente envia à DHT uma requisição de busca pelo conteúdo. Como explicado no Capítulo 6, em primeira instância a requisição em questão é recebida pelo DNS, que a encaminha para um dos *peers* da DHT; e

4. **Acessar o conteúdo**: Esta etapa é realizada automaticamente pelo navegador e, desta forma, torna-se transparente ao usuário. Durante a fase de acesso ao conteúdo,

o fluxo é direcionado para o Provedor de Dados, que apresenta ao usuário (através do navegador) as informações e o conteúdo do objeto.

É importante deixar claro que – com exceção da etapa 4 – as funcionalidades fornecidas pelo navegador devem ser requisitadas pelo usuário, que interage diretamente com o componente.

7.2.2 Provedor de Serviços

Pode-se dizer que Provedores de Serviço funcionam como *containers* de metadados em uma federação de Bibliotecas Digitais OAI e, desta forma, estão aptos à fornecer interfaces de busca sobre os metadados colhidos.

Neste sentido, nos experimentos realizados, SPs disponibilizam duas funcionalidades principais:

1. **Busca por Título**: Oferece ao usuário a possibilidade de realizar buscas a partir do título do objeto. A busca é executada sobre a base de metadados do SP e retorna o identificador do objeto (DOI) requerido. Vale destacar que a execução desta funcionalidade é requisitada pelo componente navegador através da mensagem `SEARCH`; e

2. **Fornecimento de Metadados**: Conforme exposto na Seção 6.2.3, na arquitetura proposta SPs implementam o verbo `GetRecord`, a fim de disponibilizar os metadados colhidos ao Provedores de Dados. Neste sentido, ao receber a mensagem `GETRECORD`, juntamente com o DOI do objeto, SPs entregam ao DP o metadado requerido.

A base de metadados utilizada pelo SP é armazenada em um documento XML, seguindo o formato apresentado na Figura 7.1. Uma vez que não se faz necessário a utilização de todos os atributos descritos no padrão *Dublin Core* para validação da arquitetura, nos experimentos apresentados metadados são compostos apenas pelo título, formato do conteúdo e identificador do objeto.

Ainda é importante salientar que, durante os experimentos não existe necessidade de validar a colheita de metadados por parte do SP, uma vez que este serviço teve seu

```
<?xml version="1.0" encoding="UTF-8"?>
<record_list>
  <record>
    <title>Uma Arquitetura OAI para Preservação Digital utilizando Redes
          Peer-to-Peer Estruturadas</title>
    <format>postscript</format>
    <identifier>http://pdht.c3sl.ufpr.br/1884/10000</identifier>
  </record>
</record_list>
```

Figura 7.1: Exemplo de documento XML utilizado como base de metadados nos Provedores de Serviço

funcionamento previamente comprovado e vêm sendo utilizado com sucesso por diversas aplicações que implementam o protocolo OAI-PMH. Neste sentido, assume-se que a base de metadados utilizada pelo SP foi manualmente e preenchida.

7.2.3 Sistema de Nomes de Domínios

O componente que "simula" o funcionamento de um Sistema de Nomes de Domínios (DNS) também foi implementado em Java, e tem como principal atribuição a escolha do nodo da DHT que irá receber a requisição de busca.

Para tal, assume-se que o DNS possui uma lista pré-definida dos possíveis *peers* disponíveis na rede. A partir desta lista, o componente realiza – de forma automática – verificações periódicas sobre os nodos da DHT, a fim de identificar os nodos indisponíveis.

Neste sentido, a disponibilidade dos nodos da DHT é verificada através do envio da mensagem `OK?` para cada um dos *peers*, conforme ilustrado na Figura 7.2. Após o envio, o DNS aguarda a mensagem `OK` como resposta. Caso a conexão com um *peer* não seja estabelecida ou respondida no tempo limite, o DNS considera esse *peer* inativo, e o retira temporariamente da lista de nodos que podem receber uma requisição de busca.

A Figura 7.2 apresenta um exemplo meramente ilustrativo, onde o *peer* identificado como `Nodo1` não responde o pedido de atividade requisitado pelo DNS e, desta forma, é considerado inativo. Neste contexto, observa-se o acesso do navegador ao `Nodo2`, que é considerado ativo pelo componente DNS. É importante ficar claro que durante os experimentos realizados, ao receber uma requisição de acesso do navegador, o DNS seleciona

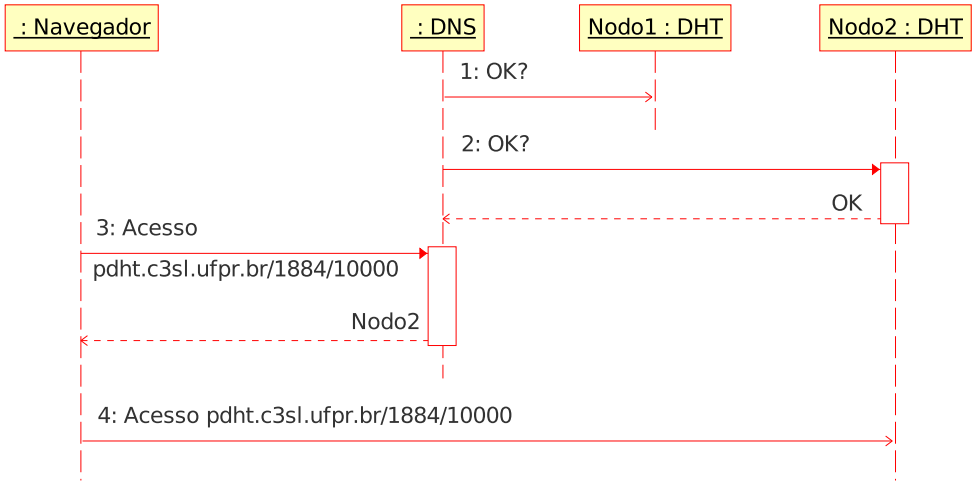

Figura 7.2: Exemplo de troca de mensagens entre DNS e DHT para manter uma lista de nodos ativos no sistema

aleatoriamente – e de forma transparente ao usuário – um dos *peers* em sua lista de nodos ativos.

Para ilustrar na prática a troca de mensagens entre o DNS e a DHT, a Figura 7.3 apresenta partes de um *log* registrado pelo componente DNS durante os testes de identificação de atividade dos nodos da DHT.

```
(a) Mensagem 'OK?' enviado para [macalan.c3sl.ufpr.br/200.17.202.6:2100]
    Resposta 'OK' recebida de [macalan.c3sl.ufpr.br/200.17.202.6:2100]

(b) Mensagem 'OK?' enviado para [montecristo.c3sl.ufpr.br/200.17.202.51:2100]
    Servidor nao respondeu no tempo limite. Conexao terminada pelo cliente
```

Figura 7.3: Registro de *log* da troca de mensagens entre o DNS e a DHT

Neste exemplo, o *log* representado pela letra `(a)` apresenta a o funcionamento correto da comunicação, onde o *peer* requisitado responde ao DNS com a mensagem `OK`. Já no *log* `(b)`, observa-se que o acesso ao servidor *montecristo.c3sl.ufpr.br* não pôde ser realizado. Assim, este servidor é considerado inativo até que volte a responder as requisições do DNS normalmente.

7.2.4 Tabela *Hash* Distribuída

Tendo em vista que o objetivo deste trabalho é concepção de uma arquitetura para preservação digital, a implementação do componente DHT ocorreu de forma a suprir

as necessidades de validação dos fluxos propostos.

Neste sentido, a DHT é vista pelo sistema como uma *caixa preta*, com seu funcionamento interno projetado de forma simples, e tendo como principal objetivo atender requisições de busca e inserção de objetos.

Assim, é assumido que cada um dos *peers* integrantes da DHT possui uma lista, armazenada em XML, que identifica todos os nodos da rede. A comunicação entre os nodos ocorre de forma direta, como ilustra a Figura 7.4 (a).

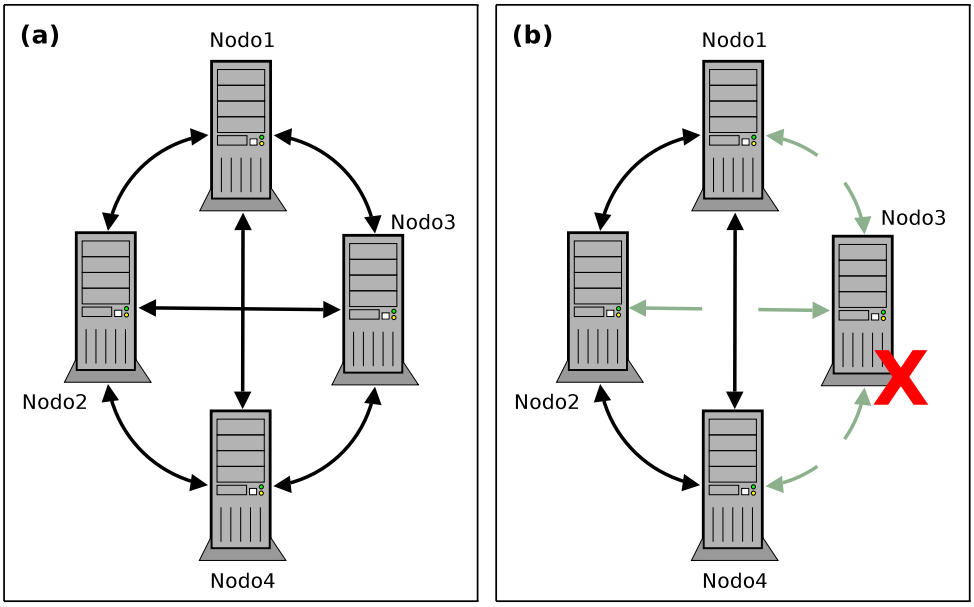

Figura 7.4: Comunicação interna entre os *peers* da DHT

Além disso, considera-se que nodos da DHT podem estar indisponíveis, como no exemplo apresentado na Figura 7.4 (b). O exemplo demonstra uma rede composta por 4 *peers*, sendo que um deles (`Nodo3`) encontra-se inativo, ou seja, não recebe e nem envia mensagens. Neste cenário, a comunicação entre os outros *peers* transcorre normalmente, o que garante o funcionamento do sistema e torna possível o acesso à réplicas de objetos.

É de suma importância deixar claro que durante os experimentos não é considerado a entrada de novos *peers* na rede. No entanto, como dito anteriormente, nodos inicialmente inseridos podem entrar e sair do sistema.

Neste contexto, buscas são iniciadas pela DHT a partir do recebimento do DOI como mensagem e retornam a URI do DP onde o objeto foi localizado. Vale destacar que durante o processo de busca, réplicas do conteúdo requisitado podem ser encontradas em

qualquer um dos nodos ativos.

Ademais, inserções são requisitadas através do envio da mensagem `INSERT` pelo DP. Juntamente com esta mensagem, a DHT recebe o DOI, o conteúdo do arquivo e o *fator de confiança* objeto. Tais valores são utilizados como parâmetros durante o processo de inserção, como apresentado na Figura 6.7.

Deste modo, objetos são armazenados no sistema de arquivos, respeitando a estrutura de diretórios apresentada na Seção 6.2.1.1. Além disso, o protótipo da DHT utiliza-se de um documento XML que armazena os identificadores de objetos e é utilizado como referência para o conteúdo durante o processo de busca.

Por fim, é interessante ressaltar que a replicação de conteúdo é realizada de forma transparente ao usuário. Para descobrir o número de réplicas necessárias para o objeto, a DHT considera que a *confiabilidade do sítio* é de 80% para cada um de seus nodos. Assim, o objeto é replicado até que sua *confiabilidade local* seja maior ou igual ao *fator de confiança* fornecido pelo usuário (um exemplo de cálculo de confiabilidade e a definição dos termos utilizados foram apresentados na Seção 3.4).

7.2.5 Provedor de Dados

Na arquitetura proposta, Provedores de Dados possuem duas principais atribuições: (i) inserir novos objetos no sistema; e (ii) fornecer acesso ao conteúdo para os usuários. Neste sentido, ambas funções são detalhadas a seguir.

Inserção de Objetos:

Para inserir um novo objeto no sistema, o administrador acessa o componente navegador e preenche as informações do objeto. Inicialmente, são fornecidos o título e o formato do arquivo, utilizados posteriormente para a criação do metadado. Em seguida, o conteúdo do objeto e o *fator de confiança* são informados. Por fim, o navegador envia ao DP a mensagem `INSERT`, juntamente com as informações coletadas.

Com posse dessas informações, inicia-se o processo de criação do identificador do objeto digital (DOI). Para tal, assume-se que o DP obtém seu identificador e a URI que endereça a

DHT, a partir de um arquivo de configuração. Além disso, um número seqüencial é criado e utilizado na composição do DOI. Assim, obtém-se o identificador do objeto respeitando o padrão apresentado na Seção 2.1 (por exemplo, *http://pdht.c3sl.ufpr.br/1884/10000*).

O passo seguinte é a criação do metadado do objeto. Da mesma forma que ocorre nos Provedores de Serviço, metadados são armazenados – de maneira simplificada – em documentos XML (vide Figura 7.1) e possuem os atributos *title*, *format* e *identifier*, que armazenam respectivamente o título, o formato e o identificador do objeto.

Por fim, para que a inserção seja efetivada, o DP envia à DHT a mensagem `INSERT`, juntamente com o DOI, o conteúdo e o *fator de confiança*.

Fornecimento de Acesso ao Conteúdo:

Conforme exposto anteriormente, o acesso ao conteúdo é requisitado ao DP pelo componente navegador. Ao receber a requisição, o DP procura localmente pelo objeto e seu respectivo metadado.

É importante observar que metadados não são replicados, então, podem não ser encontrados na base local do Provedor de Dados. Assim, para apresentar as informações do objeto ao usuário, o DP requisita o metadado a um Provedor de Serviços (endereços dos Provedores de Serviço são armazenados no arquivo de configuração), utilizando-se da mensagem `GETRECORD`. A requisição é realizada através do componente Harvester de metadados.

Para finalizar, com base no metadado e no conteúdo do objeto, o DP formata uma resposta e a envia ao navegador, para que o usuário tenha acesso ao conteúdo.

7.3 Testes de Validação da Arquitetura

Nas seções subseqüentes são apresentados os testes realizados com os protótipos descritos anteriormente, com o intuito de validar os protocolos propostos na arquitetura. Os testes ocorrem sobre um ambiente pré-definido, composto por um DNS, um Provedor de Serviços e quatro *peers* que desempenham o papel de DHT e Provedor de Dados.

Todos os testes foram executados em cinco computadores (cohiba, dalmore, macalan, montecristo, talisker), hospedados no Departamento de Informática, utilizados concorrentemente pelos alunos e professores dos cursos de Ciências Exatas da Universidade Federal do Paraná. Durante a realização dos experimentos, tais computadores possuiam em média uma configuração de *4GB* de Memória de Acesso Aleatório (RAM) e processadores *Dual Core* de *2GHz*.

Vale ressaltar que todas as requisições no ambiente são iniciadas pelo componente navegador.

7.3.1 Teste de Inserção de Conteúdo

O objetivo do teste de inserção é validar o fluxo de mensagens apresentado na Seção 6.3.1, através da Figura 6.7. O experimento é iniciado com o preenchimento das informações do objeto, utilizando-se do componente navegador, conforme descreve a Seção 7.2.5.

A Figura 7.5 apresenta a interface fornecida pelo navegador durante o processo de inserção. Observando a figura nota-se claramente o preenchimento dos campos requeridos, bem como uma mensagem de `sucesso`, informando ao administrador que o objeto foi corretamente inserido.

```
rufino@talisker:~/experimentos$ java -jar navegador.jar insert
Iniciando Componente Navegador em 200.17.202.57
Preencha o titulo do objeto: Dissertacao Everton
Preencha o formato do objeto: postscript
Preencha o conteudo do objeto: dissertacao-everton.ps
Preencha o fator de confianca: 0.98

Conectar-se ao Provedor de Dados em: macalan.c3sl.ufpr.br
Navegador [22:04:10][200.17.202.57]: Conectado ao servidor [macalan.c3sl.ufpr.br/200.17.202.6:2400]
Navegador [22:04:10][200.17.202.57]: Enviando pedido de insercao. Aguarde...
Navegador [22:04:10][200.17.202.57]: Insercao realizada com SUCESSO!
```

Figura 7.5: Interface do navegador durante o processo de inserção

Embora a Figura 7.5 apresente o resultado final do processo, é importante salientar que antes da mensagem de `sucesso` ser apresentada ao administrador, o protocolo de inserção de conteúdo proposto pela arquitetura foi executado internamente, e de forma transparente ao usuário.

Com o intuito de criar um artefato visual para facilitar a validação do protocolo, a Figura 7.6 apresenta, de forma ordenada, o fluxo de mensagens entre os componentes que participaram do processo de inserção deste experimento. Vale destacar que o rastreamento das mensagens é realizado através de um arquivo de *log*, mantido por cada um dos protótipos, que registra as mensagens enviadas e recebidas.

Observando a Figura 7.6 percebe-se que inicialmente o navegador envia ao DP uma requisição de inserção através da mensagem `INSERT`. Em conjunto são enviadas as informações descritivas do objeto. Na etapa seguinte, o Provedor de Dados realiza a criação do DOI e do metadado e, por fim envia à DHT a mensagem `INSERT`, para que a inserção e preservação do objeto no sistema sejam efetivadas.

```
Navegador [22:04:10][200.17.202.57]: Conectado ao servidor [macalan.c3sl.ufpr.br/200.17.202.6:2400]
Navegador [22:04:10][200.17.202.57]: Enviando pedido de insercao. Aguarde...

DP [22:04:10][200.17.202.6:2400]: Pedido de INSERCAO recebido de [talisker.c3sl.ufpr.br/200.17.202.57:60035]
DP [22:04:10][200.17.202.6:2400]: Capturando dados para criar DOI
DP [22:04:10][200.17.202.6:2400]: DOI criado: http://pdht.c3sl.ufpr.br/100/4
DP [22:04:10][200.17.202.6:2400]: Criando metadado
DP [22:04:10][200.17.202.6:2400]: Metadado criado com sucesso
DP [22:04:10][200.17.202.6:2400]: Enviando pedido de INSERT para DHT

DHT [22:04:10][200.17.202.6:2300]: Pedido de INSERCAO recebido de [macalan.c3sl.ufpr.br/200.17.202.6:39757]
DHT [22:04:10][200.17.202.6:2300]: Executando insert()
DHT [22:04:10][200.17.202.6:2300]: Objeto http://pdht.c3sl.ufpr.br/100/4 inserido com sucesso
DHT [22:04:10][200.17.202.6:2300]: Criando 2 REPLICAS

DHT [22:04:10][200.17.202.6:2300]: Requisitanto INSERCAO para [cohiba.c3sl.ufpr.br/200.17.202.52:2300]
DHT [22:04:10][200.17.202.52:2300]: Pedido de INSERCAO recebido de [macalan.c3sl.ufpr.br/200.17.202.6:45533]
DHT [22:04:10][200.17.202.52:2300]: Executando insert()
DHT [22:04:10][200.17.202.52:2300]: Objeto http://pdht.c3sl.ufpr.br/100/4 inserido com sucesso
DHT [22:04:10][200.17.202.52:2300]: Enviando mensagem sucesso para [macalan.c3sl.ufpr.br/200.17.202.6:45533]

DHT [22:04:10][200.17.202.6:2300]: Requisitanto INSERCAO para [dalmore.c3sl.ufpr.br/200.17.202.56:2300]
DHT [22:04:10][200.17.202.56:2300]: Pedido de INSERCAO recebido de [macalan.c3sl.ufpr.br/200.17.202.6:60634]
DHT [22:04:10][200.17.202.56:2300]: Executando insert()
DHT [22:04:10][200.17.202.56:2300]: Objeto http://pdht.c3sl.ufpr.br/100/4 inserido com sucesso
DHT [22:04:10][200.17.202.56:2300]: Enviando mensagem sucesso para [macalan.c3sl.ufpr.br/200.17.202.6:60634]

DHT [22:04:10][200.17.202.6:2300]: Enviando mensagem sucesso para [macalan.c3sl.ufpr.br/200.17.202.6:39757]
DP [22:04:10][200.17.202.6:2400]: Enviando mensagem sucesso para [talisker.c3sl.ufpr.br/200.17.202.57:60035]
Navegador [22:04:10][200.17.202.57]: Insercao realizada com SUCESSO!
```

Figura 7.6: Mensagens de *log* ordenadas para ilustrar o fluxo de inserção

Ao receber a mensagem INSERT, a DHT dá início ao processo de inserção do objeto através da invocação do método `insert(name, value, reliability)`. Durante o procedimento, réplicas do objeto são criadas até que o *fator de confiança* definido pelo usuário (98%) seja alcançado. Neste experimento, 3 cópias do objeto foram inseridas,

sendo duas delas réplicas.

Com base nas asserções e resultados obtidos, considera-se que o protocolo de inserção de objetos proposto pela arquitetura é adequado ao contexto apresentado neste trabalho, uma vez que foi possível observar o funcionamento correto de todo o fluxo, bem como a replicação de conteúdo, durante a execução do processo.

7.3.2 Teste de Busca de Conteúdo

Seguindo a vertente apresentada durante o teste de inserção (Seção 7.3.1), o objetivo do teste de busca de conteúdo é validar o fluxo de mensagens apresentado na Seção 6.3.2 e ilustrado através da Figura 6.8.

Neste sentido, o experimento é dividido em duas partes: (i) resgate do identificador do objeto; e (ii) busca pelo conteúdo no sistema.

Durante a execução da etapa (i), o identificador do objeto é recuperado a partir de uma consulta ao Provedor de Serviços. A busca é requisita pelo componente navegador através do envio da mensagem `SEARCH` e do título do objeto, como pode ser observado na Figura 7.7 (a).

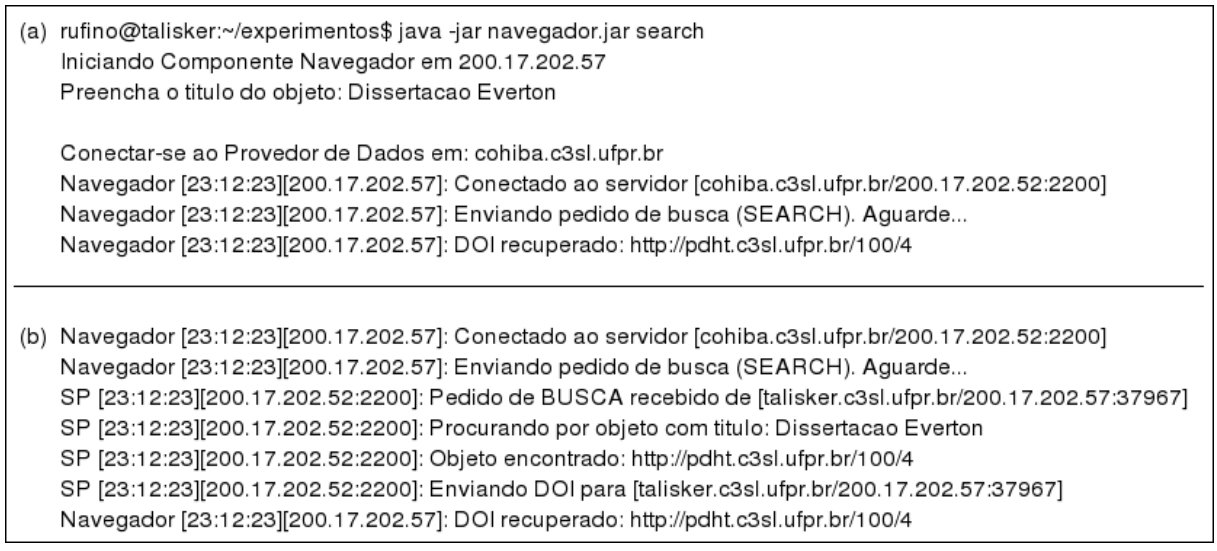

Figura 7.7: (a) Interface de busca do navegador; (b) Fluxo de mensagens entre o navegador e o Provedor de Serviços.

O fluxo de mensagens entre o componente navegador e o SP é apresentada em detalhes na Figura 7.7 (b). Conforme dito anteriormente, SPs mantêm localmente – em formato

XML – uma base de metadados dos objetos. Assim, ao receber a mensagem `SEARCH` o SP consulta sua base local, a fim de encontrar o identificador do objeto solicitado e atender a requisição do usuário.

Uma vez que o DOI foi obtido inicia-se a segunda etapa do experimento, que é caracterizada pela solicitação de acesso ao conteúdo no Sistema de Preservação Digital. Nesta etapa, o componente navegador requisita à DHT o acesso ao objeto, utilizando-se do DOI (recuperado na etapa anterior) informado pelo usuário, e como resposta obtém as informações e o conteúdo do objeto (vide Figura 7.8).

```
rufino@talisker:~/experimentos$ java -jar navegador.jar access
Iniciando Componente Navegador em 200.17.202.57
Preencha o identificador do objeto: http://pdht.c3sl.ufpr.br/100/4

Navegador [02:34:13][200.17.202.57]: Conectado ao servidor [cohiba.c3sl.ufpr.br/200.17.202.52:2000]
Navegador [02:34:13][200.17.202.57]: Resposta do DNS: macalan.c3sl.ufpr.br
Navegador [02:34:13][200.17.202.57]: Conectado ao servidor [macalan.c3sl.ufpr.br/200.17.202.6:2300]
Navegador [02:34:13][200.17.202.57]: Requisitando acesso ao objeto a DHT
Navegador [02:34:13][200.17.202.57]: Objeto encontrado, acessando objeto no DP
Navegador [02:34:14][200.17.202.57]: Conectado ao servidor [macalan.c3sl.ufpr.br/200.17.202.6:2400]
Navegador [02:34:14][200.17.202.57]: Objeto recuperado:
Navegador [02:34:14][200.17.202.57]: Titulo do objeto: Dissertacao Everton
Navegador [02:34:14][200.17.202.57]: file:/home/ppginf/rufino/experimentos/macalan/100/4/4.ps
Navegador [02:34:14][200.17.202.57]: Arquivo armazenado no diretorio corrente
```

Figura 7.8: Interface do navegador durante o processo de busca/acesso a um objeto

Em primeira instância, uma mensagem contendo o DOI é encaminhada ao DNS, que decide qual *peer* da DHT será responsável por receber a requisição. Vale destacar que nodos são selecionados de forma aleatória, respeitando a lista de *peers* em atividade, conforme descreve a Seção 7.2.3.

Para que seja possível realizar a validação completa do fluxo apresentado na Figura 6.8, dois diferentes testes de busca são executados. O primeiro consiste, de maneira geral, em recuperar do sistema a cópia original do objeto, enquanto a obtenção de uma réplica é o objetivo traçado para o segundo teste. A seguir, ambos os testes são descritos em detalhes.

Teste de obtenção da Cópia Original:

Durante a execução deste teste assume-se que todos os protótipos utilizados estão ativos, no entanto, é considerado que o componente DNS endereça apenas o nodo em que o conteúdo solicitado foi originado. Tal asserção é possível através da alteração da lista – utilizada pelo DNS – que contém os nodos que integram a rede.

Neste contexto – que gera a impossibilidade de obtenção de réplicas do conteúdo – ao acessar o DOI (obtido anteriormente) o componente navegador é direcionado pelo DNS ao nodo da DHT que originou o registro solicitado. Após localizar o conteúdo, o nodo da DHT responde ao navegador a URI que identifica o conteúdo no Provedor de Dados, como apresenta a Figura 7.9

```
Navegador [02:34:13][200.17.202.57]: Conectado ao servidor [cohiba.c3sl.ufpr.br/200.17.202.52:2000]

DNS [02:34:13][200.17.202.52]: Requisicao recebida de [talisker.c3sl.ufpr.br/200.17.202.57:50002]
DNS [02:34:13][200.17.202.52]: Fazendo busca na lista de nodos ativos
DNS [02:34:13][200.17.202.52]: Selecionado nodo macalan.c3sl.ufpr.br
DNS [02:34:13][200.17.202.52]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:50002]

Navegador [02:34:13][200.17.202.57]: Resposta do DNS: macalan.c3sl.ufpr.br
Navegador [02:34:13][200.17.202.57]: Conectado ao servidor [macalan.c3sl.ufpr.br/200.17.202.6:2300]
Navegador [02:34:13][200.17.202.57]: Requisitando acesso ao objeto a DHT

DHT [02:34:13][200.17.202.6:2300]: Pedido de BUSCA recebido de [talisker.c3sl.ufpr.br/200.17.202.57:60253]
DHT [02:34:13][200.17.202.6:2300]: Executando lookup()
DHT [02:34:13][200.17.202.6:2300]: Objeto macalan.c3sl.ufpr.br/100/4 encontrado
DHT [02:34:13][200.17.202.6:2300]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:60253]

Navegador [02:34:13][200.17.202.57]: Objeto encontrado, acessando objeto ao DP
Navegador [02:34:14][200.17.202.57]: Conectado ao servidor [macalan.c3sl.ufpr.br/200.17.202.6:2400]

DP [02:34:14][200.17.202.6:2400]: Pedido de ACESSO recebido de [talisker.c3sl.ufpr.br/200.17.202.57:39755]
DP [02:34:14][200.17.202.6:2400]: Recuperando metadado do objeto
DP [02:34:14][200.17.202.6:2400]: Formatando mensagem de resposta
DP [02:34:14][200.17.202.6:2400]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:39755]

Navegador [02:34:14][200.17.202.57]: Objeto recuperado:
Navegador [02:34:14][200.17.202.57]: Titulo do objeto: Dissertacao Everton
Navegador [02:34:14][200.17.202.57]: file:/home/ppginf/rufino/experimentos/macalan/100/4/4.ps
Navegador [02:34:14][200.17.202.57]: Arquivo armazenado no diretorio corrente
```

Figura 7.9: Mensagens de *log* ordenadas para ilustrar o fluxo de acesso à cópia original do objeto

Para que o conteúdo seja recuperado, o navegador requisita automaticamente ao DP o acesso ao objeto. Por fim, o DP localiza localmente o metadado do objeto e formata uma mensagem de resposta. Com o intuito de facilitar os testes assume-se que além

das informações descritivas do objeto, o DP envia ao navegador o conteúdo do arquivo, simulando assim uma operação do *download*.

Teste de obtenção de uma Réplica:

Para possibilitar a realização do teste de obtenção de uma réplica, a lista dos *peers* conhecidos pelo componente DNS foi preenchida com todos os nodos da rede. Entretanto, considera-se neste teste que o *peer* que originou o objeto requisitado (*macalan.c3sl.ufpr.br*) encontra-se inativo, que inviabiliza a obtenção da cópia original do objeto.

```
Navegador [02:41:22][200.17.202.57]: Conectado ao servidor [cohiba.c3sl.ufpr.br/200.17.202.52:2000]

DNS [02:41:22][200.17.202.52]: Requisicao recebida de [talisker.c3sl.ufpr.br/200.17.202.57:60031]
DNS [02:41:22][200.17.202.52]: Fazendo busca na lista de nodos ativos
DNS [02:41:22][200.17.202.52]: Selecionado nodo talisker.c3sl.ufpr.br
DNS [02:41:22][200.17.202.52]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:60031]

Navegador [02:41:22][200.17.202.57]: Resposta do DNS: talisker.c3sl.ufpr.br
Navegador [02:41:22][200.17.202.57]: Conectado ao servidor [talisker.c3sl.ufpr.br/200.17.202.57:2300]
Navegador [02:41:22][200.17.202.57]: Requisitando acesso ao objeto a DHT

DHT [02:41:22][200.17.202.57:2300]: Pedido de BUSCA recebido de [talisker.c3sl.ufpr.br/200.17.202.57:39625]
DHT [02:41:22][200.17.202.57:2300]: Executando lookup()
DHT [02:41:22][200.17.202.57:2300]: Objeto nao encontrado, requisitando busca a dalmore.c3sl.ufpr.br
DHT [02:41:22][200.17.202.57:2300]: Conectado ao servidor [dalmore.c3sl.ufpr.br/200.17.202.56:2300]
DHT [02:41:22][200.17.202.56:2300]: Pedido de BUSCA recebido de [talisker.c3sl.ufpr.br/200.17.202.57:20623]
DHT [02:41:22][200.17.202.56:2300]: Objeto dalmore.c3sl.ufpr.br/100/4 encontrado
DHT [02:41:22][200.17.202.56:2300]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:20623]
DHT [02:41:22][200.17.202.57:2300]: Objeto encontrado em dalmore.c3sl.ufpr.br
DHT [02:41:23][200.17.202.57:2300]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:39625]

Navegador [02:41:23][200.17.202.57]: Objeto encontrado, acessando objeto ao DP
Navegador [02:41:23][200.17.202.57]: Conectado ao servidor [dalmore.c3sl.ufpr.br/200.17.202.56:2400]

DP [02:41:23][200.17.202.56:2400]: Pedido de ACESSO recebido de [talisker.c3sl.ufpr.br/200.17.202.57:50106]
DP [02:41:23][200.17.202.56:2400]: Recuperando metadado do objeto
DP [02:41:23][200.17.202.56:2400]: Requisitando metadado (GETRECORD) ao SP
DP [02:41:23][200.17.202.56:2400]: Conectado ao servidor [cohiba.c3sl.ufpr.br/200.17.202.52:2000]

SP [02:41:23][200.17.202.52:2000]: GETRECORD recebido de [dalmore.c3sl.ufpr.br/200.17.202.56:40367]
SP [02:41:23][200.17.202.52:2000]: Recuperando Metadado solicitado
SP [02:41:23][200.17.202.52:2000]: Enviando resposta para [dalmore.c3sl.ufpr.br/200.17.202.56:40367]

DP [02:41:23][200.17.202.56:2400]: Metadado recebido do SP
DP [02:41:23][200.17.202.56:2400]: Formatando mensagem de resposta
DP [02:41:23][200.17.202.56:2400]: Enviando resposta para [talisker.c3sl.ufpr.br/200.17.202.57:50106]

Navegador [02:41:23][200.17.202.57]: Objeto recuperado:
Navegador [02:41:23][200.17.202.57]: Titulo do objeto: Dissertacao Everton
Navegador [02:41:23][200.17.202.57]: file:/home/ppginf/rufino/experimentos/dalmore/100/4/4.ps
Navegador [02:41:23][200.17.202.57]: Arquivo armazenado no diretorio corrente
```

Figura 7.10: Mensagens de *log* ordenadas para ilustrar o fluxo de acesso a uma réplica do objeto

Diante do contexto apresentado, as trocas de mensagens ocorrem de forma similar ao apresentado no teste anterior. No entanto, é possível observar, através da Figura 7.10, a comunicação entre os nodos da DHT durante a busca; e a utilização do verbo `GETRECORD` pelo DP, com o intuito de obter o metadado do Provedor de Serviços.

Neste sentido, o componente navegador acessa o DOI e é direcionado pelo DNS à um dos nodos da DHT, que realiza a busca por toda rede até encontrar o objeto solicitado. A DHT então envia ao DP a URI que identifica o objeto localmente. Considerando que o DP não contém os metadados do registro, este envia a mensagem `GETRECORD` ao SP e, após receber os metadados, formata um resposta e apresenta ao usuário.

Com base nos resultados obtidos, é possível observar que o objeto solicitado pelo usuário foi recuperado com sucesso mesmo diante de uma falha no nodo que o originou. Assim, é possível afirmar que o comportamento dos protótipos transcorreu conforme previsto, o que valida os protocolos apresentados pela arquitetura proposta nesse trabalho.

7.4 Considerações Finais

Neste capítulo foram apresentados os experimentos realizados com o intuito de validar o fluxo de mensagens, bem como o comportamento de cada um dos componentes utilizados na proposta da arquitetura.

Considera-se que, mesmo com a implementação de protótipos não funcionais, os resultados obtidos vão de encontro à proposta inicial e, desta forma, validam a arquitetura. Sendo assim, pode-se dizer que a utilização de sistemas de preservação baseados na arquitetura proposta aumenta, de forma considerável, a confiabilidade e disponibilidade dos objetos digitais.

Finalizando, é importante deixar claro que aspectos de desempenho não foram considerados durante os experimentos.

CAPÍTULO 8

CONCLUSÕES E TRABALHOS FUTUROS

Bibliotecas são as entidades responsáveis pelo compartilhamento e preservação da informação e exercem um papel fundamental para o funcionamento e crescimento de uma sociedade. A constante evolução tecnológica faz com que cada vez mais bibliotecas digitais assumam importância similar às bibliotecas tradicionais, no entanto, oferecendo acesso à coleções de objetos a partir do meio digital.

Neste contexto, o protocolo OAI-PMH é considerado uma ferramenta indispensável para facilitar o acesso ao conteúdo digital, uma vez que ele fornece um padrão para disseminação e compartilhamento de metadados, possibilitando a busca pela informação a partir de diversos Provedores de Serviço ou motores de busca na Internet.

É importante observar que, da mesma forma que ocorre no mundo real, é extremamente necessário manter o conteúdo digital preservado, ou seja, garantir sua integridade diante problemas inesperados ou falhas catastróficas, a fim de assegurar o acesso à informação para futuras gerações.

Assim, o trabalho desenvolvido nesta dissertação vem firmar a importância das bibliotecas digitais com entidades responsáveis pelo compartilhamento de conhecimento, e contribuir para o constante crescimento da utilização do protocolo OAI-PMH como padrão para a disseminação de acesso ao conteúdo, através da partilha de metadados. Além disso, a concepção de um mecanismo de preservação digital garante a confiabilidade e disponibilidade das informações por longos períodos de tempo (décadas ou séculos).

Do ponto de vista técnico, a arquitetura para preservação digital proposta neste trabalho especifica um conjunto de componentes – bem como define protocolos para comunicação entre eles – que possibilitam a replicação, auditoria, e busca de objetos digitais. Além disso, a arquitetura foi projetada para integrar-se à federações de bibliotecas digitais OAI, oferecendo total compatibilidade com o protocolo OAI-PMH.

Uma vez que a arquitetura proposta baseia-se em redes P2P, considera-se sua utilização apropriada para a concepção de bibliotecas digitais preservadas em larga escala e com baixo custo. Se comparada a outros mecanismos de preservação digital, observa-se que a utilização de um *fator de confiança* em nossa arquitetura, que identifica a importância do objeto, cria um ambiente propício para a otimização do sistema, uma vez que apenas o conteúdo necessário é replicado, evitando a "poluição" da rede com réplicas desnecessárias.

A validação da arquitetura através da implementação de protótipos, proporcionou uma visualização detalhada do comportamento dos componentes e das trocas de mensagens propostas. Os experimentos realizados comprovam que é possível desenvolver um sistema de preservação baseado na arquitetura, que proporcione o real aumento de confiabilidade e disponibilidade dos objetos digitais.

Por fim, é importante destacar que por ter como base uma Tabela *Hash* Distribuída (DHT) – que indexa separadamente cada um dos objetos – a arquitetura é mais indicada para concepção de sistemas que disponibilizam ítens individuais de conteúdo, como por exemplo Bibliotecas Digitais de Teses e Dissertações.

8.1 Contribuições

De maneira geral, a principal contribuição deste trabalho foi a definição de uma arquitetura para preservação digital, com o objetivo de aumentar a confiabilidade e disponibilidade dos objetos disponíveis em Bibliotecas Digitais OAI.

Dentre as contribuições especificas do trabalho realizado pode-se citar:

1. Concepção de uma nova abordagem para preservação digital de conteúdo sobre bibliotecas digitais OAI;

2. Definição geral de uma tabela *hash* distribuída que utiliza um *fator de confiança* para cada objeto digital;

3. Criação de uma arquitetura baseada nessa abordagem;

4. Validação do funcionamento da arquitetura através da realização de experimentos;

5. Implementação de protótipos que permitiram a definição e validação da arquitetura.

8.2 Trabalhos Futuros

Como perspectiva de trabalhos futuros, pretende-se definir, de forma detalhada, e implementar a tabela *hash* distribuída proposta neste trabalho, a fim de garantir a confiabilidade do conteúdo.

Além disso, em futuras versões da arquitetura, é de extrema importância considerar a propriedade de auto-adaptação dos sistemas P2P. Tal propriedade possibilita a reorganização das réplicas durante entrada e saída de nodos do sistema, fazendo com que o número mínimo de réplicas necessárias para garantir a preservação do objeto esteja disponível.

Por fim, objetiva-se a criação de um mecanismo de auto-recuperação, ou seja, que possibilite a reconstituição do conteúdo de um nodo que tenha perdido seus objetos. Para tal, o nodo em questão pode recuperar seus metadados a partir do Provedor de Serviços (aproveitando da característica de bilateralidade, proposta neste trabalho) e requerer seus objetos aos outros nodos da DHT.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] arXiv.org Website. Disponivel em: www.arxiv.org/. Acesso em jul. 2008.

[2] Dspace website. Disponivel em: www.dspace.org/. Acesso em jul. 2008.

[3] Dublin Core Metadata Initiative (DCMI). Disponivel em: dublincore.org/. Acesso em mai. 2008.

[4] The eMule-Project. Disponivel em: www.emule-project.net/. Acesso em abr. 2008.

[5] Eprints for Digital Repositories. Disponivel em: www.eprints.org/. Acesso em jul. 2008.

[6] Extensible Markup Language (XML). Disponivel em: www.w3.org/XML. Acesso em mar. 2008.

[7] The Freenet Project. Disponivel em: http://freenetproject.org/. Acesso em abr. 2008.

[8] Genome@home website. Disponivel em: http://genomeathome.stanford.edu/. Acesso em abr. 2008.

[9] Gnutella website. Disponivel em: www.gnutella.com/. Acesso em abr. 2008.

[10] Kazaa website. Disponivel em: www.kazaa.com/. Acesso em mai. 2008.

[11] The LOCKSS Project Website. Disponivel em: www.lockss.org/. Acesso em jun. 2008.

[12] Open Archives Initiative. Disponivel em: www.openarchives.org. Acesso em jun. 2008.

[13] SciELO – Scientific Electronic Library Online. Disponivel em: http://www.scielo.org/. Acesso em jul. 2008.

[14] SETI@home website. Disponivel em: http://setiathome.berkeley.edu/. Acesso em abr. 2008.

[15] The Digital Object Identifier System. Disponivel em: www.doi.org/. Acesso em jun. 2008.

[16] The Handle System. Disponivel em: www.handle.net/. Acesso em jun. 2008.

[17] Visionary Technology in Library Solutions. Disponivel em: www.vtls.com/. Acesso em jun. 2008.

[18] W3C HyperText Markup Language (HTML) Home Page. Disponivel em: www.w3.org/html/. Acesso em mar. 2008.

[19] Maristella Agosti, Hans-Jörg Schek, e Can Türker, editors. *Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Pre-proceedings of the Sixth Thematic Workshop of the EU Network of Excellence DELOS, S. Margherita di Pula, Cagliari, Italy, 24-25 June, 2004*. Edizioni Libreria Progetto, Padova, 2004.

[20] Benjamin Ahlborn. OAI-P2P: A Peer-to-Peer Network for Open Archives. *ICPPW '02: Proceedings of the 2002 International Conference on Parallel Processing Workshops*, páginas 462, Washington, DC, USA, 2002. IEEE Computer Society.

[21] A. Amrou, K Maly, e M. Zubair. Freelib: Peer-to-peer-based Digital Libraries. *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, páginas 9–14, Washington, DC, USA, 2006. IEEE Computer Society.

[22] Stephanos Androutsellis-Theotokis e Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, dezembro de 2004.

[23] Mary Baker, Mehul Shah, David S. H. Rosenthal, Mema Roussopoulos, Petros Maniatis, TJ Giuli, e Prashanth Bungale. A fresh look at the reliability of long-term

digital storage. *EuroSys '06: Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, páginas 221–234, New York, NY, USA, 2006. ACM.

[24] M. Nelson C. Lagoze, H. Van de Sompel e S. Warner. The Open Archives Initiative Protocol for Metadata Harvesting, version 2.0. Disponivel em: www.openarchives.org/OAI/openarchivesprotocol.html. Acesso em jul. 2008.

[25] Brian Cooper, Arturo Crespo, e Hector Garcia-Molina. Implementing a reliable digital object archive. *ECDL '00: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, páginas 128–143, London, UK, 2000. Springer-Verlag.

[26] Brian Cooper e Hector Garcia. Creating Trading Networks of Digital Archives. *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, páginas 353–362, New York, NY, USA, 2001. ACM.

[27] Brian F. Cooper e Hector Garcia-Molina. Peer-to-peer data trading to preserve information. *ACM Trans. Inf. Syst.*, 20(2):133–170, 2002.

[28] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, e Ion Stoica. Wide-area cooperative storage with CFS. *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, outubro de 2001.

[29] Consultative Committee for Space Data Systems (CCSDS). Reference Model for an Open Archival Information System (OAIS), note = Disponivel em: http://ssdoo.gsfc.nasa.gov/nost/isoas/. Acesso em mai. de 2008., year = 2003.

[30] Ali Ghodsi, Luc Onana Alima, e Seif Haridi. Symmetric replication for structured peer-to-peer systems. *DBISP2P*, páginas 74–85, 2005.

[31] Stewart Granger. Emulation as a digital preservation strategy. *D-Lib Magazine*, 6(10), 2000.

[32] Steve Hitchcock, Tim Brody, Jessie M. N. Hey, e Leslie Carr. Digital Preservation Service Provider Models for Institutional Repositories: Towards Distributed Services. *D-Lib Magazine*, 13(5/6), 2007.

[33] R. Huebsch, J. M. Hellerstein, N. L. Boon, T. Loo, S. Shenker, e I. Stoica. Querying the internet with pier. *In Proceedings of 19th International Conference on Very Large Databases (VLDB)*, setembro de 2003.

[34] S. Balusani A. Mathur S. Sudeep K. Maly, M. Zubair e W. Wolters. DL-COTF: an XML based digital library for U. S. Navy's operational test and evaluation force... *in Proceedings of the 2002 ACM symposium on Applied computing table of contents*, páginas 493–497, Madrid, Spain, 2002. ACM Press.

[35] Roger I. Khazan. Group membership: a novel approach and the first single-round algorithm. *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, páginas 347–356, New York, NY, USA, 2004. ACM.

[36] Predrag Knezevic, Andreas Wombacher, e Thomas Risse. Enabling high data availability in a dht. *dexa*, 00:363–367, 2005.

[37] Salma Ktari, Mathieu Zoubert, Artur Hecker, e Houda Labiod. Performance evaluation of replication strategies in dhts under churn. *MUM '07: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, páginas 90–97, New York, NY, USA, 2007. ACM.

[38] Carl Lagoze e Herbert Van de Sompel. The Open Archives Initiative: Building a low-barrier interoperability framework. *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, páginas 54–62, New York, NY, USA, 2001. ACM Press.

[39] Maohua Lu e Tzi cker Chiueh. Challenges of long-term digital archiving: A survey. *Technical Report ECSL-TR-205*, 2006.

[40] Siu Man Lui e Sai Ho Kwok. Interoperability of peer-to-peer file sharing protocols. *SIGecom Exch.*, 3(3):25–33, 2002.

[41] Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, e Mary Baker. The LOCKSS peer-to-peer Digital Preservation System. *ACM Trans. Comput. Syst.*, 23(1):2–50, 2005.

[42] Vidal Martins, Esther Pacitti, e Patrick Valduriez. Survey of data replication in p2p systems. 2007.

[43] P. Mockapetris e K. J. Dunlap. Development of the domain name system. *SIGCOMM Comput. Commun. Rev.*, 18(4):123–133, 1988.

[44] A. Philip, G. Fausto, K. Anastasios, M. John, S. Luciano, e Z. Ilya. Data management for peer-to-peer computing: A vision. *In Proceedings of the Workshop on the Web and Databases (WebDB'02)*, 2002.

[45] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, e Scott Schenker. A scalable content-addressable network. *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, páginas 161–172, New York, NY, USA, 2001. ACM Press.

[46] Vicky Reich e David S.H. Rosenthal. LOCKSS: A Permanent Web Publishing and Access System. *D-Lib Magazine*, 7(6), 2001.

[47] David S. H. Rosenthal e Vicky Reich. Permanent Web Publishing. *USENIX Annual Technical Conference, FREENIX Track*, páginas 129–140. USENIX, 2000.

[48] David S. H. Rosenthal, Thomas Robertson, Thomas Lipkis, Vicky Reich, e Seth Morabito. Requirements for digital preservation systems: A bottom-up approach. *CoRR*, abs/cs/0509018, 2005.

[49] Antony Rowstron e Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–??, 2001.

[50] Yasushi Saito e Marc Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.

[51] Karthikeyan Sankaralingam, Simha Sethumadhavan, e James C. Browne. Distributed pagerank for p2p systems. *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, páginas 58, Washington, DC, USA, 2003. IEEE Computer Society.

[52] Clay Shirky. What is P2P... and what isn't? Disponivel em: www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html. Acesso em 11 abr. 2008.

[53] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, e Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *In Proceedings of the ACM Special Interest Group on Data Communications (SIGCOMM) '01 Conference*, San Diego, California, agosto de 2001.

[54] C. Meghini R. Hecht T. Risse, P. Knezevic e F. Basile. The bricks infrastructure - an overview. *In Proc. of 75th Conference on Electronic Imaging, the Visual Arts & Beyond (EVA 2005)*, Moscow, Russia, 2005.

[55] Kenneth Thibodeau. Overview of technological approaches to digital preservation and challenges in coming years. *The State of Digital Preservation: An International Perspective*, Washington D.C., 2002.

[56] Patrick Valduriez e E. Pacitti. Data management in large-scale p2p systems. *Int. Conf. on High Performance Computing for Computational Science (VecPar'2004)*, páginas 109–122. LNCS 3402, Springer, 2004.

[57] Hakim Weatherspoon e John Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, páginas 328–338, London, UK, 2002. Springer-Verlag.

[58] C. Webb. Guidelines for the preservation of digital heritage, 2003.

[59] Joel M. Winett. Request for Comment (RFC) 147 – The Definition of a Socket. Relatório técnico, Lincoln Laboratory, 1971. Disponível em: http://tools.ietf.org/html/rfc147.

[60] Yanfei XU. A P2P Based Personal Digital Library for Community. *PDCAT '05: Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, páginas 796–800, Washington, DC, USA, 2005. IEEE Computer Society.